VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Bakalářská práce

2013 **Ondřej Mazal**

VŠB - Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a biomedicínského inženýrství

Měřicí a řídicí komponenty pro sběrnici LIN
Measuring and Control Components for LIN Bus

Ostrava, 2013 **Ondřej Mazal**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Ondřej Mazal**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2601R004 Měřicí a řídicí technika

Téma: Měřicí a řídicí komponenty pro sběrnici LIN
Measuring and Control Components for LIN Bus

Zásady pro vypracování:

1. Obecný popis sběrnice LIN.
2. Návrh a realizace master jednotky pro komunikaci s PC na platformě Freescale.
3. Návrh a realizace slave jednotek pro řízení ss motoru a krokového motoru.
4. Návrh a realizace slave jednotky s LCD displejem.
5. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] ŠVIMBERSKÝ, Zdeněk. *LIN - automobilová sběrnice.* Praha, 2007. Bakalářská práce. ČVUT v Praze, Fakulta elektrotechnicky, Katedra řídicí techniky.
[2] OTEVŘEL,Vít. *Využití LIN sběrnice v řídicích systémech.* Ostrava, 2010. Bakalářská práce. VŠB-Technická univerzita Ostrava, Fakulta elektrotechniky a informatiky, Katedra měřicí a řídicí techniky.
[3] ACARNELY, Paul. *Stepping Motors: A Guide to Theory and Practice (IEE Control Engineering Series 63).* 4th ed. London(UK): the Institution of Electrical Engineers, 2002. 176 s. ISBN 085296417X/978-0852964170.
[4] *LIN consortium.* [online] [cit. 2000-09-11]. Dostupné z: http://www.lin-subbus.de .

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Zdeněk Slanina, Ph.D.**

Datum zadání: 19.11.2010

Datum odevzdání: 07.05.2013

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne 7. května 2013

Ondřej Mazal

Poděkování

Děkuji především svému vedoucímu bakalářské práce Ing. Zdeňkovi Slaninovy, PhD., za jeho cenné rady a připomínky. Také bych chtěl poděkovat dalším učitelům, doktorandům a svým kolegům, kteří mi poskytli cenné rady pro úspěšnou realizaci této práce. Také bych rád poděkoval všem, kteří mě podporovali v průběhu realizace této práce.

Abstrakt

Tato bakalářská práce se zabývá rozborem a užitím měřících a řídících komponent využívající LIN sběrnici. Jde o jednoduchou a levnou průmyslovou datovou sběrnici používanou převážně v automobilech pro ovládání prvků nekritického charakteru jako nastavení zrcátek, stahování oken atd. Cílem práce je seznámení a práce se sběrnicí LIN, poté realizovaní modulů pro distribuované řízení a jejich následné programování a uživatelská vizualizace pro PC a nakonec analýza celkově dosažených výsledků.

Klíčová slova

LIN (Local Interconnect Network), MCU (mikrokontrolér, Freescale, Motorola, sběrnice, master, slave, PWM, USB, driver.

Abstract

Main goal of this thesis is control component analysis and usage on LIN bus.LIN bus is simple and cheap data bus used for automotive industry. In cases of noncritical character like window control or car´s mirror sliding, ect.). Goal of this thesis is introduction and work with LIN bus, after creating this module for distributed control and programming. Last task in this thesis is to make user visualisation for PC and result analysis.

Key words

LIN (Local Interconnect Network), MCU (microcontroller), Freescale, Motorola, bus, master, slave, PWM, USB, driver.

Seznam použitých symbolů a zkratek

BDM	Background Debug Module, konektor pro programování/debugování
CAN	Controller Area Network
DPS	Deska plošneho spoje
GND	Ground, zem
ID	Identifikátor
IDE	Integrated Development Environment
LCD	Liquid Crystal Display
LIN	Local Interconnect Network
MCU	Mikrokontrolér
OOP	Objektově orientované programování
SCI	Serial Communication Interface
SLIC	Slave LIN Interface Controller
UART	Universal Asynchronnous Receiver Transmiter
USB	Seriové rozhraní

Seznam obrázků

Obsah

1. Úvod

Úkolem bakalářské práce je obeznámení s LIN sběrnicí a jejím užitím pro měřící a řídící systémy. Prvním úkolem je obeznámení se s LIN sběrnicí pomocí doporučené literatury a zjistit její obecné specifikace jako napěťové úrovně užívané touto sběrnicí a jak realizovat komunikaci přes sběrnici atd. druhý úkol zní, vytvořit řídící obvod s mikrokontrolérem na platformě FREESCALE, zároveň obsahující LIN rozhraní pro možnost připojení slave zařízení a po sběrnici USB pro komunikaci s PC. Třetím úkolem je vytvořením slave modulů, které obsahují kromě LIN a MCU rozhraní ss motor řízený pomocí PWM modulace a druhého řízeného modulu jež obsahuje krokový motor čtvrtý krok je vytvoření modulu obsahující LCD display a realizace komunikace mezi PC a master jednotkou což zahrnuje vytvoření firmwaru do mikrokontroléru firmy freescale, které jsou užity jak v master i v slave jednotce a program pro obsluhu a řízení motorů.

1.1 Struktura práce

Bakalářská práce je rozčleněna do 7 samostatných kapitol, ve kterých se bude podrobně pojednávat o problematice tohoto tématu práce a následné realizaci řešení této práce.

Struktura kapitol je rozdělena následovně:

1. *Úvod*
2. *Rozbor LIN (Local interconnect network)*
3. *Návrh a realizace master modulů.*
4. *Návrh a realizace slave modulu s LCD.*
5. *Návrh a realizace modulů se ss a krokovým motorkem.*
6. *Zhodnocení výsledků.*

2. LIN (Local interconnect Network)

Specifikace LIN sběrnice 1.0 byla vytvořena v červenci roku 1999, u vytvoření této sběrnice stálo LIN sdružení , které se v jádru skládá ze sedmi firem a to: Audi AG,BMW AG, Daimler AG, Motorola (Freescale Halbletioer Deutshlad GmbH),Volcano Communications Technologies, Volkswagen AG a Volvo car corporation. První užití LIN sběrnice se datuje v roc 2001, po deseti letech se od prvního užití LIN 1.0 již existuje specifikace LIN 2.2 její předchozí specifikace 2.1 byla vytvořena na podzim roku 2001. Jde o otevřený standart sériové automobilové sběrnice, který je koncipován jako single master/multiple slave jež předpokládá užití 3 – 10 jednotek s LIN u jednoho vozu. Typické užití LINu představují řízení oken, zámků, zrcátek, klimatizace, stěračů a různých aplikovaných snímačů atd. Původní navržení LINu byl původně navržen jako doplněk sběrnice CAN, nemá za cíl nahradit CAN sběrnici ,která je rychlá a robustní ,ale spíše pokrýt okruh aplikací kde je CAN zbytečným luxusem, nebo zatím nebyly napojený na elektronickou řídící jednotku vozu. Cena která je investována na propojení s lokální sítí automobilu je 2 až 3 krát nižší než v případě užití CANu. Celková částka se zdá býti zanedbatelná vzhledem k celkové ceně vozu ,ale v případě sériové výroby jež se v případě automobilového průmyslu pohybuje kolem desítek až stovek tisíců už tak zanedbatelná není. Pro realizaci komponent s LIN není třeba speciálních driverů, nároky uspokojí i běžný mikrokontrolér s obvodem UART/SCI nebo v případě jednotek SLAVE i bez tohoto obvodu, jednotky slave nevyžadují přesné a drahé krystalové oscilátory, požadavky uspokojí i levné RC oscilátory. LIN je v dnešní době podporován nejen automobilovými koncerny (mimo jmenovaných také Toyota, MAN,Renault,Opel) ale i společnostmi zabývající se i elektroprůmyslem např. Fujitsu, Toshiba Electronics Europe GmbH,Texas Instruments Deutschland GmbH, Agilent Technologies...), je to otevřený standart, to znamená, že protokol je volně k dostání pro zájemce skrze sdružení LIN.

Struktura sítě LIN je vybudována na bázi jednoho modulu Master a několika modulů slave, který je limitován počtem identifikátorů a fyzickým vlastnostem sběrnice.[1][2][7]

LIN se skládá s těchto částí:

1. mikroprocesor obsahující události, vysílající a přijímající data na sběrnici LIN
2. Obvod UART/SCI jako HW řadič sériové komunikace (převážně je součástí jednočipového mikrokontroléru) nebo softwarový driver komunikace LIN
3. LIN transceiver (driver LIN) realizující fyzickou vrstvu protokolu LIN (převod na TTL úrovně do fyzické vrstvy LIN – napěťová úroveň 0 až 12) [1][2][7]

2.1 Základní vlastnosti

LIN je otevřený standard sériové automobilové sběrnice třídy A, skládá se z jedné master jednotky a z jedné nebo více jednotek slave vycházející ze specifikace single-master / multiple-slave, maximum připojených slave jendotek je však 16. jde o jednovodičovou sériovou sběrnici, která využívá pro sériový přenos dat formátu UART/SCI.[1][2][7]

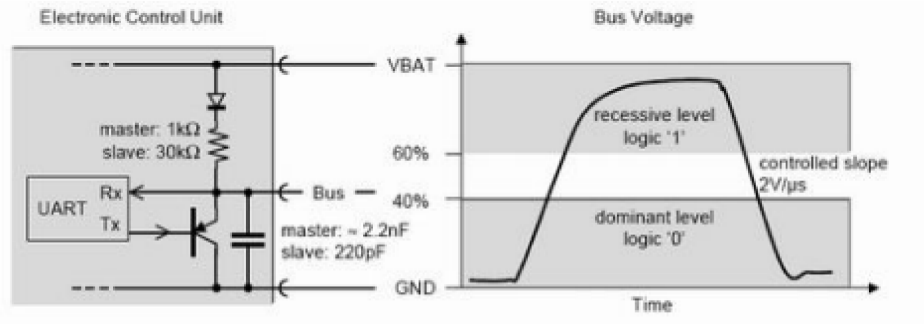
- rychlost komunikace po sběrnici je 2400 b/s až 19200 kb/s
- Jednovodičová sběrnice s 12 V úrovní
- Komunikace po sběrnici je typu master – slave
- lze propojit navzájem až 17 modulů (1 x master, 16 x slave)
- v přenášené zpravě je možno přenést 0,2,4 nebo 8 bajtů
- Zakončovací rezistor:
  1. $1\ k\Omega$ pro master jednotku

2. 30 $k\,\Omega$ pro slave jednotku

- nedoporučuje se přesahovat délku sběrnice 40 m
- koncepce driverů sběrnice vychází ze standardu ISO 9141 s vylepšeními v oblasti EMC
- zabezpečení dat provedeno kontrolním součtem
- hlavička zabezpečena dvojicí paritních bitů

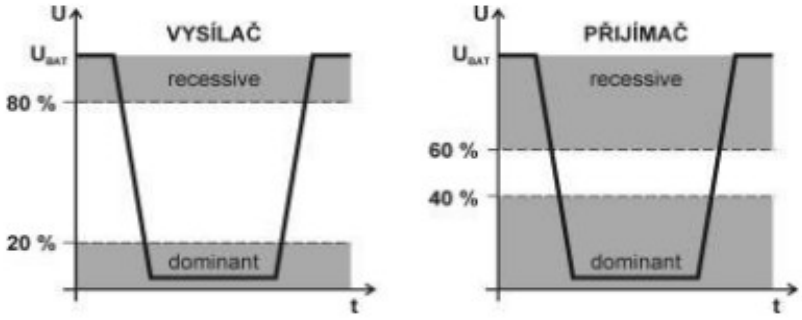
2.2 Fyzická vrstva

Je odvozena ze standardu ISO 9141 tento standard je vyvinut k diagnostickým účelům pro použití v servisech. Je zřejmě že pokud má být fyzická vrstva tohoto standardu užita pro vozy, musí být navrhnuty určité změny, např. Strmost náběžných a sestupných hran je záměrně omezena s ohledem na minimalizaci množství vyžadovaného rušení a je pamatováno i na posun zemního potenciálu a různé poruchy . Princip sběrnice LIN je v použití jednoho vodiče pro obousměrnou komunikaci pomocí realizace funkce logického součinu prostřednictvím spínačů a odporů zapojených na LIN sběrnici v každém připojeném zařízení. Jsou definovány dvě vzájemné komplementární hodnoty stavů na sběrnici a to dominant a recessive. Velikost a rozsah jednotlivých úrovní jsou vztaženy vzhledem k palubnímu napětí automobilu generovaného akumulátorem (autobaterií) 12 V a nebo alternátorem a to až do 14.5 V.[1][2][7]



Obr. 1: Fyzická vrstva [2]

Spínače při sepnutí spojují sběrnici se zemí. Stačí, aby byl sepnut alespoň jeden a sběrnice přejde do stavu dominant, což představuje stav logické nuly rezistory zapojené mezi napájecí napětí a sběrnici pak na ní udržují, pokud není žádný spínač sepnutý stav resessive tedy logickou jedničku. Sběrnici zakončuje pull-up rezistor, v případě master jednotky je to hodnota 1 $k\,\Omega$ a u slave 30 $k\,\Omega$,které udržují tyto stavy. Pro případ poruchy zdroje je zde v sérii s rezistorem zapojena dioda. Aby bylo zařízení odolnější proti elektromagnetickému rušení, připojují se paralerně k pinům driveru LIN kondezátory.[1][2][7]



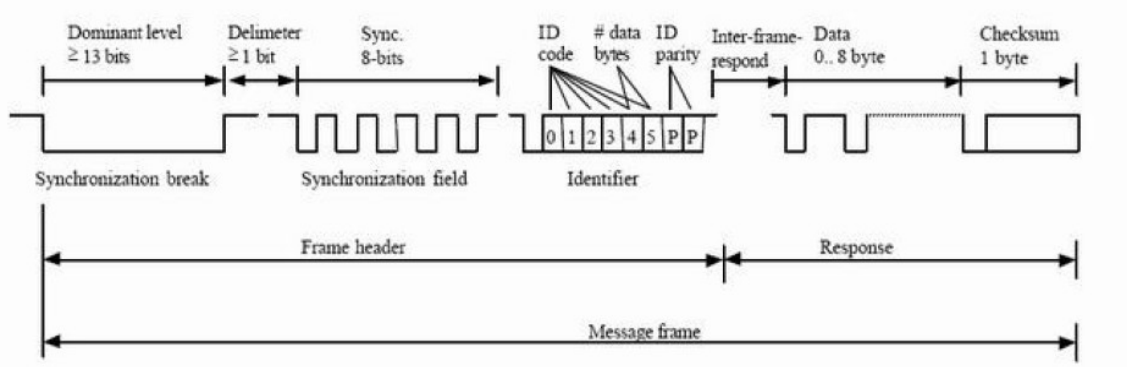
Obr. 2: Rozhodovací úrovně vysílače a přijímače [2]

2.3 Linková vrstva

LIN protokol se zakládá na UART prokolu, tzn. že zprávy jsou kódovány po bajtech. Hlavní odlišnost oproti UARTu je v zavedení synchronizační pauzy, Pomocí LIN zprávy odeslané Masterem je možno synchronizovat, za adresovat slave jednotky a přenos dat mezi Master a Slave moduly. Celý přenos je samozřejmě řízen master jednotkou. LIN zpráva se skládá ze dvou menších rámců, které odděluje mezirámcová mezerou s variabilní délkou.[1][2]

- Hlavička (Header frame nebo Command frame) – vysílá pouze Master
- Odpověď (Response frame nebo Data frame) – vysílá Master i Slave

Header se dále skládá ze tří částí a to ze synchronizační pauzy (synchronization break), synchronizační pole (sychronization field) a identifikátor (Identifier).



Obr. 3: Formát rámce zprávy [1]

2.3.1 Synchronizační pauza (Synchronization Break)

Synchronizační pauza je dlouhá minimálně 13 nulových bitů, tato pauza je vytvořena pro spolehlivou detekci zprávy odeslané na sběrnici slave moduly.[1][2][7]

2.3.2 Synchronizační pole

Toto pole slouží k synchronizaci hodin slave jednotek s master jednotkou, vždy před přijetím nové zprávy. Proto zcela postačí když přesný čas užívá pouze master. Slave jednotky jsou synchronizovány tak, že od sestupné hrany start bitu až po pátou sestupnou hranu synchronizačního bajtu změří čas a podělí ho 8, tím je získán baud time rate master jednotky.[1][2][7]

2.3.3 Identifikátor

Hlavička obsahuje i identifikační pole, které se ještě děli na 6 bitový identifikátor a 2 bity zbývají na paritu. Teoreticky tedy dostaneme až 64 identifikatorů. Čtvrtý a pátý bit ID pole určuje kolik dat se bude přenášet v datové oblasti (0, 2, 4, nebo 8 bajtů). Uzly nemají žádnou fyzickou adresu, např. (MAC), místo toho je v paměti předefinován seznam platných identifikátorů.[1][2][7]

2.3.4. Datový rámec

Datový rámec se sestává se z 0 až 8mi bajtů, délka je závislá na hardwarové a softwarové realizovaných ve všech uzlech, datový rámec také obsahuje 1 byte kontrolního součtu, který je počítám z dat přenášených po sběrnici.[1][2][7]

2.4 Komunikace

v rámci komunikace existují 2 úrovně priority master a slave:

1. **Master:***[1]*

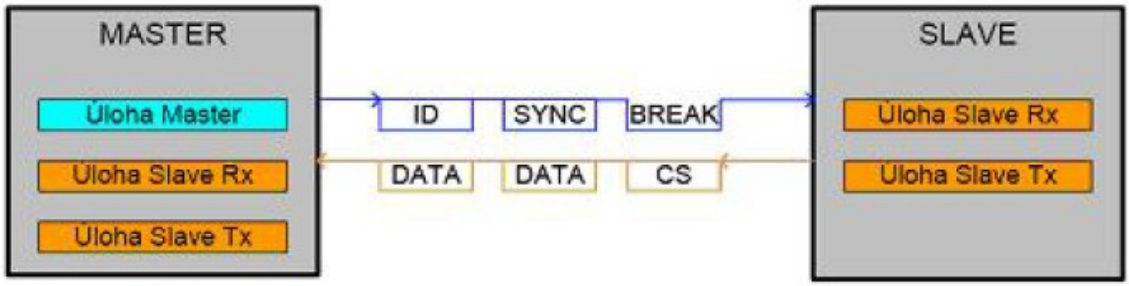
- řídí veškerou komunikaci po sběrnici
- definuje přenosovou rychlost
- vysílá header a jemu náležící části (synchronizační pauzu,synch. pole, identifikátor)
- monitoruje a potvrzuje data pomocí kontrolního součtu
- přepíná do slave jednotky do sleepmodu a a probouzí je
- reaguje na wake-up signál od slave jednotek

2. Slave:*[1]*

- čeká na synchronizační impulz
- synchronizuje se dle synch. Pole
- na základě stavu v identifikátoru vykonává:
- nečinnost
- přijímá informace
- vysílá informace
- kontroluje nebo zasílá kontrolní součet

2.4.1 Způsoby zasílání dat Master to Slave

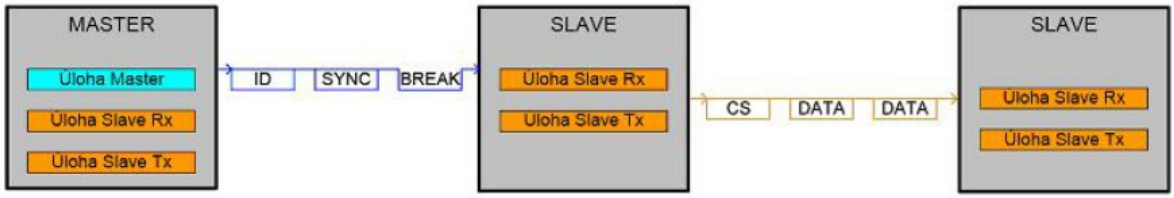
Je to typický příklad řízení akčních členů Slave (v tomto případě zapínání/vypínání motorku, předávání dat na LCD display k zobrazení atd.). V tomto případě odesílání dat z MASTER do Slave se užije ve Slave modulu TX. Master zašle header s ID, na kterou si odpoví sám tím že odešle data, která přijme Slave. [1][2][7]



Obr. 4: Komunikace Master to Slave [2]

2.4.2 Způsoby zasílání dat Slave to Slave

Jde o variantu přímé komunikace mezi jednotkami Slave, kde takto snižujeme vytížení jednotky Master a dá se takto řešit čtení dat nebo ovládání akčních členů, jelikož by Master nejdříve přijal data sám a poté by je teprve odvysílal data pro Slave. V tomto případě Master vyšle header na který odpoví Slave vysláním dat která ovšem místo do Master budou odeslána na jiný Slave jelikož tyto data nejsou pro Master prioritní. [1][2][7]



Obr. 5: Komunikace Slave to Slave [2]

2.4.3. Sleep mode

V rámci LIN specifikace existuje operace nazývaná sleep mode (režim spánku), tento režim je velice žádán v případě ,že na sběrnici dochází jen ke sporadické komunikaci, ovšem tento režim je možno aktivovat pouze z jednotky Master a to prostřednictvím řídícího rámce (command frame) s ID = 0x3C s prvním Bytem dat 0x00 vyslaným všem jednotkám. Slave jednodky se po tomto příkazu přepnou do požadovaného režimu tkzv. Nízkopříkonového (low-power mode) až do okamžiku kdy dojde k rozeslání wake-up signálu (budící signál) rozeslaný na sběrnici nebo na samostatnou jednotku

slave. Pokud by se vlivem rušení některá nebo všechny jednotky Slave nezachytily datový rámec s příkazem pro přechod na režim spánku, pak samy identifikují režim spánku po určité době nečinnosti sběrnice $T_{(Time-Out)}$ což je obvykle 4s.[1][2][7]

2.4.4 Wake-up

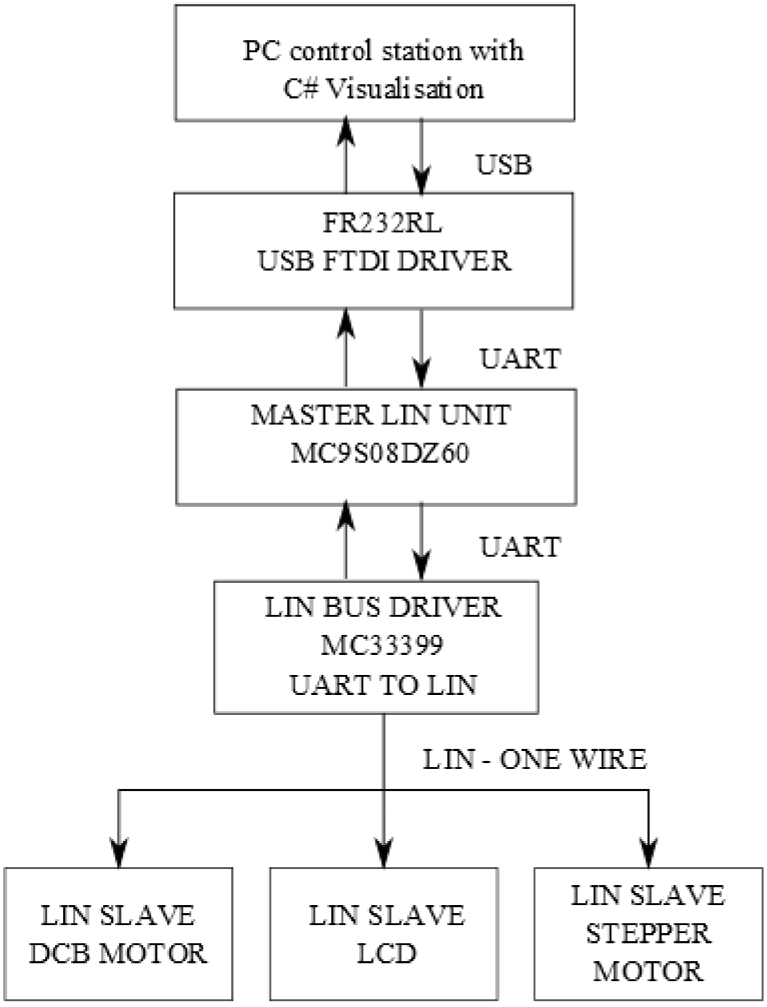
Pokud je LIN sběrnice v režimu spánku, lze sběrnici probudit vysláním signálu Wake-up z jednotky Master. Detekuje-li jednotka Slave nebo Master příkaz k probuzení a je-li nutné předat tento nový stav dalším jednotkám, bude to provedeno Wake-up signálem. Pokud není komunikační rychlost modulu Slave synchronizován s Master může být signál o 15% rychlejší nebo pomalejší, což má za následek že Master může detekovat znaky 0xC0 0x80 0x00 (7 až 9 dominantních bitů) Wake-up signál ideálně obsahuje 8 dominantních bitů a 4 recesivní jako oddělovač mezi synchronizační pauzou. Master vysílá Wake-up příkaz tolikrát dokud stále trvá požadavek na jeho zaslání, avšak při více násobného odesílání jsou vždy po zaslání prvního signálu je definována prodleva T_{T0BRK} a nedojde-li k probuzení do tří pokusů o probuzení budou signály dále zasílány s odmlkou doby T_{T3BRK} .[1][2][7]

3. Master modul

Master modul může být na sběrnici jen jeden a ten řídí komunikaci po sběrnici, jádrem zapojení je mikrokontrolér firmy Freescale MC9S08DZ60 jež je popsán níže, který obsahuje 2 SCI (sériové komunikační rozhraní). První SCI je připojeno k LIN sběrnici skrze driver MC33399 druhé SCI je užito pro komunikaci s PC skrze USB.

Důležité čipy Master modulu:

- MC33399 - LIN driver
- MC9S08DZ60ACML – mikroprocesor Master jednotky
- FT232RL – převodník UART/USB



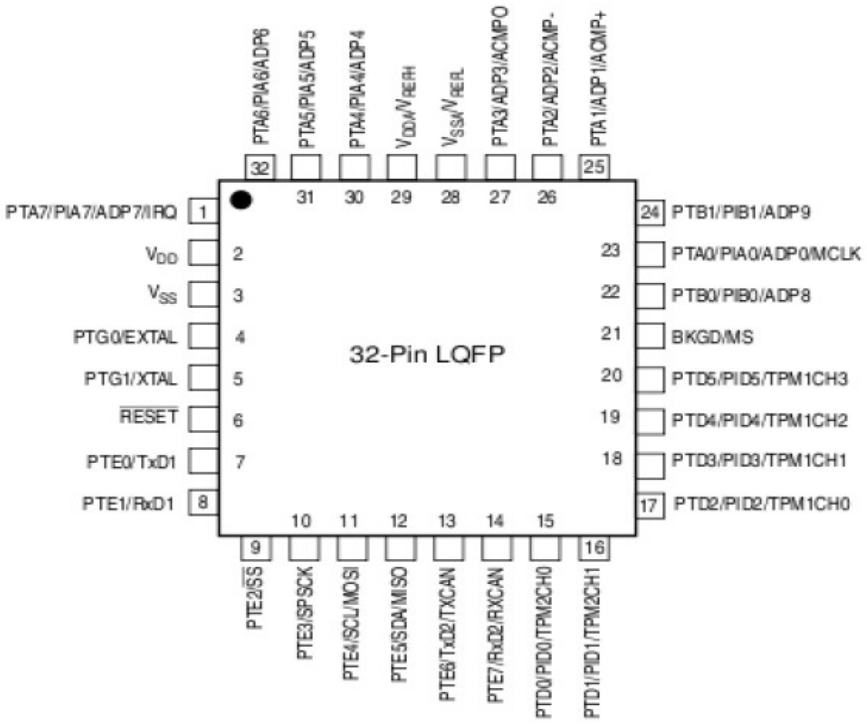
Obr. 6: Struktura aplikace Master modulu

3.1Mikrokontrolér MC9S08DZ60

Mikrokontrolér který je použit v jednotce master je 8-bitový mikrokontrolér s jádrem HCS08 jelikož byl v rámci katedry dostupný vývojový kit s tímto procesorem na kterém bylo možno ověřit funkci komunikace s počítačem a LIN sběrnici, dalším důvodem byla garantovaná podpora pro tento typ procesoru. obsahuje různé periferie a funkce, proto ho lze využít v mnoha oblastech. Hlavní oblastí využití se předpokládá v automobilovém průmyslu, protože obsahuje rozhraní CAN a LIN (SCI/UART). Hlavním důvodem proč byl zvolen tento mikrokontrolér bylo to, že má 2 SCI rozhraní. Jedním SCI bude připojeny k LIN sběrnici a pomoci druhého SCI bude možnost komunikace po USB s PC.. K tomuto účelu plně postačí 32 pinů, proto jsem zvolil pouzdro LQFP32 a přesné označeni MCU je MC9S08DZ60AMLC. Když jsou splněny všechny požadavky a jsou připojeny všechny periferie, i tak zůstane volný celý jeden port na který může být v případě potřeby a změny konstrukce přidáno jakékoliv doplňkové zařízení jako klávesnice pro kterou je v MCU vyveden řadič nebo jiné periferie. [1][3]

Základní vlastnosti *[3]*

- Napájení od 2.7 do 5.5 V.
- jádro HCS08 může běžet až na 40 MHz a vnitřní sběrnice až na 20 MHz.
- 60 kB flash.
- 4 kB RAM.
- 2 kB EEPROM.
- Můžeme připojit externí oscilátor nebo použit interní zdroj hodin (Multi-purpose Clock Generator).
- Má 26 I/O pinů.
- Má jednovodičové rozhraní pro programovaní a ladění (BDM).
- **Systémová ochrana:**
  - Computer Operating Properly (COP) Watchdog.
  - Low Voltage Detect.
  - Illegal address detection.
  - Flash block protect.
- **obsahuje tyto periferie:**
  - 1x CAN, 1x I2C, 2x SCI a 1x SPI.
  - Verze CAN protokolu je 2.0 A, B.
  - SCI podporují LIN ve verzi 2.0. Rozšířená podpora generovaní a detekce pauzy.
  - obsahuje jeden 4-kanálový, 16-bitový časovač a jeden 2-kanálový, 16-bitový časovač.
  - Obsahuje jeden 10-kanálový 12-bitový A/D převodník.
  - Má dva analogové komparátory a RTC (Real-time counter).



Obr. 7: Pouzdro LQFP32 [3]

3.1.2 SCI (Serial Communication Interface)

Jedná se o periferii jež má na starost sériovou komunikaci, přímo v tomto MCU je podpora protokolu LIN2.0, čehož je užíváno při komunikaci po LINu. Proto je SCI vybaven rozšířeními jako je generování synchronizační pauzy , v případě použití MCU jako Slave jednotky je tu možnost nastavení detekování synchronizační pauzy, jedna z dalších možností je probouzení na aktivní hranu. [1][3][5]

3.2 Popis zapojení

Jak je uvedeno výše, LIN je 12V sběrnice, protože byla primárně určena pro použití v automobilovém průmyslu. Mikroprocesor a FT232RL potřebují pro své napájení 5V, proto je na Master modulu stabilizátor napětí 7805. Před regulátorem napětí je Shottkyho dioda jež plní funkci, jako ochrana před přepólováním. Obvod 7805, do něj vstupuje 12V z LIN sběrnice a konvertuje těchto 12V na 5V pro napájení dalších obvodů na Master modulu za stabilizátorem napětí je červená LED dioda jež signalizuje zdali je připojení napájení modulu. Před LED diodou je ochranný rezistor 10k Ω . Napájení mikroprocesoru MC9S08DZ60AMLC je napájen 5V ze stabilizátoru 7805. O hodinový takt (rychlost) MCU se stará krystal 16MHz ,který je připojen na piny 4 a 5 (piny EXTAL a XTAL) mikroprocesoru. LIN driver je připojený k pinům periferie SCI1 v MCU a to 2-mi vodiči RxD a TxD. Ke druhé periferii SCI2 je připojen FT232RL opět dvěma vodiči RxD a TxD. Na pin 21 (BKGD) je připojený jediný vodič programovacího/ladícího konektoru.

LIN driver MC333999 je od firmy Freescale. Obvod je napájený 12V. Driver musí být v režimu master mezi datovým pinem LIN a napájecím pinem VSUP zapojený pull-up rezistor 10k Ω v sérii s diodou. Dále není využívána možnost uspání a probuzení proto jsou piny ENABLE (EN) a WAKE-UP

(WAKE) připojeny přímo k napěťovým úrovním které mají být na těchto pinech v normálním režimu: EN = log. 1, Wake = log. 0, nesmí se zapomenout připojit komunikační piny RxD pro příjem a TxD pro vysílání. Pin RxD je připojen k RxD pinu periferie SCI1 v MCU a pin TxD je zase připojený k pinu TxD periferie SCI v MCU.

Jednotka master má možnost připojení k PC přes sériové rozhraní USB, proto jsem užil převodník s úrovní USB FT232RL. Dále je k mikroprocesoru připojen BDM konektor, který je velmi důležitý, je to rozhraní jež se užívá pro programování a ladění (debug) programu (firmwaru) do MCU. Zapojení BDM konektoru je pro debuging vyvedený pouze jeden signál a to BKGD. Schéma a deska plošného spoje je přiložena do příloh A[1][3][5][12]

3.3 Programování jednotky master

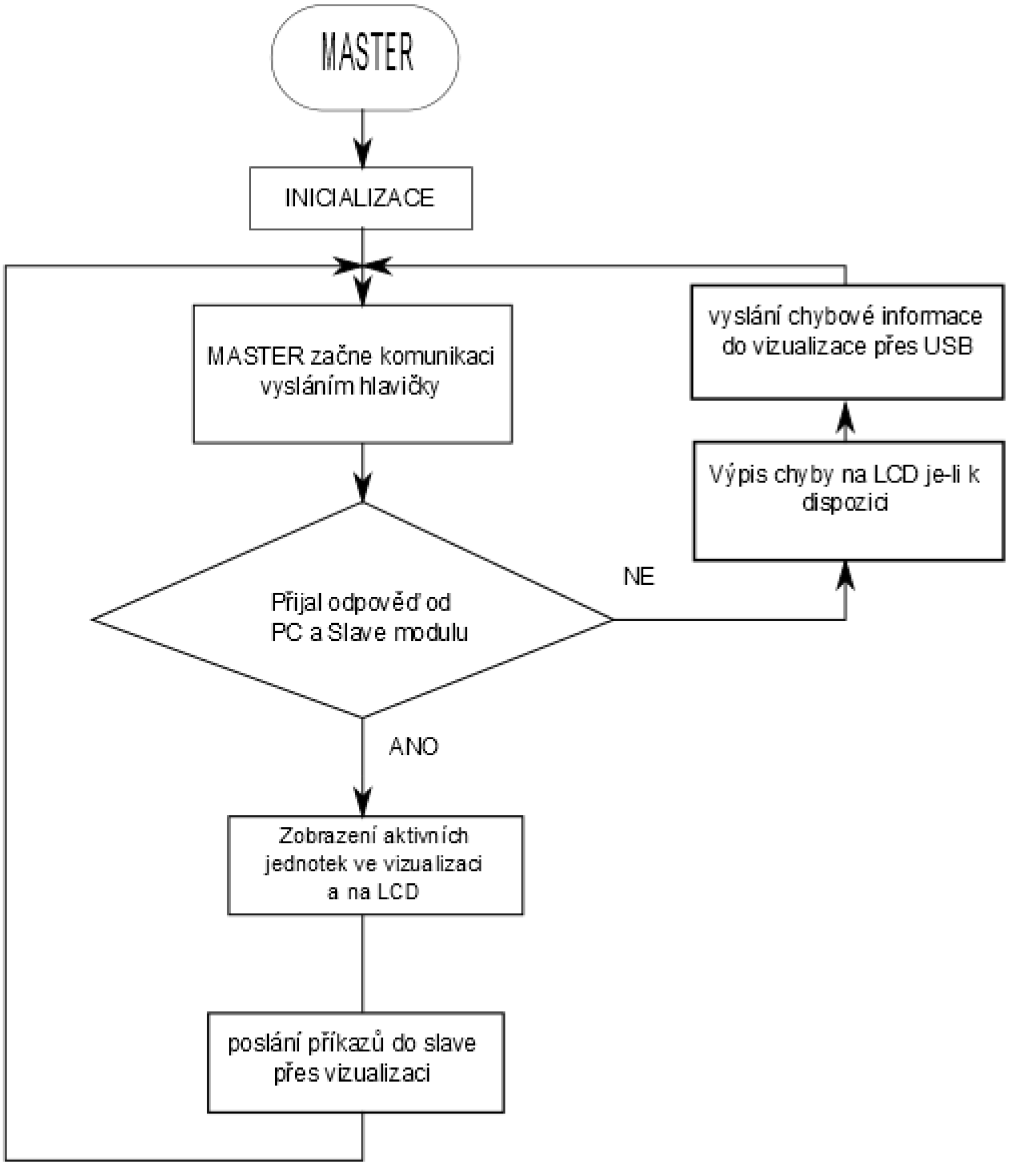
Programování samotných MCU bylo provedeno ve vývojovém prostředí Code Warrior 10.3 od firmy Freescale, jež je vylepšenou verzí freewarového prostředí Eclipse uzpůsobenou pro vývoj s plnou podporou informací o užívaných aktuálně vyráběných MCU a je pro nekomerční užití zdarma. Hlavní program je napsán v souboru *main.c.* Na začátku programu jsou umístěny headry následované definice maker, deklarací globálních proměnných a deklarací funkcí. Inicializace, která je napsána v hlavní funkci *main(),* v rámci této inicializace se provedou veškerá nastavení jako nastavení hodin MCU, periferie jako SCI1 pro LIN a SCI2 pro USB. Veškeré nastavení periferií bylo provedeno v rámci zakládání projektu v processor expertu kde byly definovány rychlostí SCI stejně jako definování typu funkce SCI zdali bude provozován jako Master nebo Slave (ale rámci programu je stále možno do samotné rychlost komunikace zasahovat), takt externího krystalu a ověření vyhovujícího nastavení MCU .

Po ukončení inicializace vstoupíme do samotné smyčky programu *for ();* V samotné smyčce jednotka užívá header LIN zprávy a očekává odpověď podřízených jednotek. Odpověď bude obsahovat tři datové bajty datové bajty o stavu samotných SLAVE jednotek, která se projeví v samotné vizualizaci indikací připojených jednotek. Pak přes vizualizaci daných příkazem přes USB je do MCU přiveden příkaz a je poslán řídící bajt na danou jednotku a provede zásah dle definovaného příkazu jako (rychlost směr zobrazení zprávy)

Header soubory užité v programu:

- *MC33399.h*
- *USB.h*
- *cekej.h*
- *main.h*
- event.h

cekej.h – obsahuje definované zpoždění pro 4 funkce definované v headru MC33399.h a pro jakékoliv přerušení má funkci cekej_ms(i) kde i je typu int a je to počet iterací při zpoždění,proto cekej_ms(2) má za následek zpoždění 2 ms.
MC33399.h – je to balíček definovaných příkazů pro obsluhu LIN sběrnice a pro identifikaci zařízení na sběrnici jelikož nové prostředí v eclipsu umožňuje přímo nastavovat SCI jako master nebo slave pro komunikaci není třeba provádět inicializaci v rámci knihovny ale stačí jen realizovat inicializaci v rámci hlavní obslužné procedury *main.h*.



Obr. 8: Vývojový diagram master aplikace

void ***LINident****(unsinged char* ***id****, unsinged char* ***delka****);* - funguje pro indentifikaci jendotlivých připojených periferních zařízení jež je možno realizovat v rámci tohoto programu 16 a jejich id je uloženo v id a delka ukládá jejich počet pro jednoduchou zprávu zařízení.

Toto je asi nejdůležitější úsek kódů jelikož je zde provedeno přečtení a kódové zamčení samotných id podřízených jednotek SLAVE.

```
void LIN_Ident(unsigned char id,unsigned char delka) {
  unsigned char P0,P1;
  id=id&0x0F;  // 0-15
    switch (delka) {
            case 2:   delka=1;break;
            case 4:   delka=2;break;
            case 8:   delka=3;break;
            default:  delka=0;break;
```

```
            }
  id=id|(delka<<4);
  P0=(id&0x01)^((id&0x02)>>1)^((id&0x04)>>3)^((id&0x10)>>4);    // (0^1^2^4)
  P1=!((id&0x01)^((id&0x04)>>2)^((id&0x10)>>4)^((id&0x20)>>5)); //~(1^3^4^5)
  id=id&0x3F;  // 0-63
  id=id|((P0&0x01)<<6);
  id=id|((P1&0x01)<<7);
  LIN_PutCharAM1(id);
}
```

void ***LIN_PosliZpravu****(unsigned char* ***id****,unsigned char* ***delka****, unsigned char* ****data)*** – je využit kanál SCI1 v režimu SCIR/TX_AM1 jež přímo definuje chod master jednotky v rámci AM funkce posílání zpráv byl zadefinován synchronizační rámec.přímo.

zde je id přiřazení samotných id k typu zařízení délka říká kolik jich je a v datech se ukládají informace do definovaného bufferu a lze zde posílat žádost o ověření o stavu zařízení data atd. Velikost bufferu je 16.

char ***LIN_VyzadejZpravu****(unsigned char* ***id****,unsigned char* ***delka****, unsigned char* ****data)*** – tento příkaz slouží pro vyžádání informací o stavu zařízení po inicializaci. Kde můžeme sledovat i stav komunikace.

char ***LIN_PrijmiZpravu****(unsigned char* ***adresa****, unsigned char* ****rxid****,unsigned char* ****rxdelka****, unsigned char* ****data)*** – přerušovací smyčka jež pracuje s daty po příjmu pokud data nedostane jen nečině vyčkává,. Při samotném přijmutí dokonce kontroluje jestli je to zpráva.

```
char LIN_PrijmiZpravu(unsigned char adresa, unsigned char *rxid,unsigned char *rxdelka,
unsigned char *data) {
        unsigned char poc,delka,id;
        unsigned int soucet=0;
        char vysledek=K_LIN_NENIZPRAVA;
        //vysledek=LIN_CekejSync();              //po proc je na prijmu
    if (vysledek==0x55) {                  //sync byte
      AM1_RecvChar(&vysledek);
     delka=((vysledek>>4)&0x03);             //pocet dat 0-3
        switch (delka) {
              case 1:   delka=2;break;
              case 2:   delka=4;break;
              case 3:   delka=8;break;
              default:  delka=0;break;
              }
     *rxdelka=delka;
     id=vysledek&0x0F;                       //id 0-15
     *rxid=id;
     if (id==adresa) {                  //-----bezny prijem-----
      for(poc=0;poc<delka;poc++) {
       __RESET_WATCHDOG(); /* feeds the dog */
       AM1_RecvChar(data+poc);
      }
```

```
    AM1_RecvChar(&vysledek);
    //LIN_KontrolaPrijmu();
    vysledek=K_LIN_JEZPRAVA;
   } else if (id==adresa+K_LIN_VYZADANI) {    // vyzadani dat
    vysledek=K_LIN_JEVYZADANI;
   }
  }
   } else if (id==adresa+K_LIN_VYZADANI) {    // vyzadani dat
    vysledek=K_LIN_JEVYZADANI;
   }
  }
  return(vysledek);
}
```

Main.h – zde probíhá samotná smyčka ve struktuře for (;;) pro samotné obslužné rutiny

USB.h – je to knihovna pro obsluhu USB kde jsou definovány veškeré funkce pro obsluhu a komunikaci, část užitého kódu vychází z dokumentace a příloh k FT232RL jež jsou uvedeny v samotném datasheetu.

void ***USB_Init****(unsigned char* ***preruseni****,unsigned int* ***rychlost****)* - je to inicializační procedura pro nastavení samotné rychlosti komunikace funkcí rychlost přes SCI a určení MASTER/SLAVE režimu.

char ***USB_GetChar****(void)* – je to procedura pro přijmutí dat když dojde k vyslání dat vyvolá přerušení a přijme data do bufferu

void ***USB_PutChar****(unsigned char* ***ch****)* – je to funkce pro přeposlání dat pro rychlosti motoru

void ***USB_PutString****(char* ****retezec****)* – je to funkce pro odeslání řetězce pro samotný LCD display pokaždé se odešle informace o zobrazení na jednom řádku.

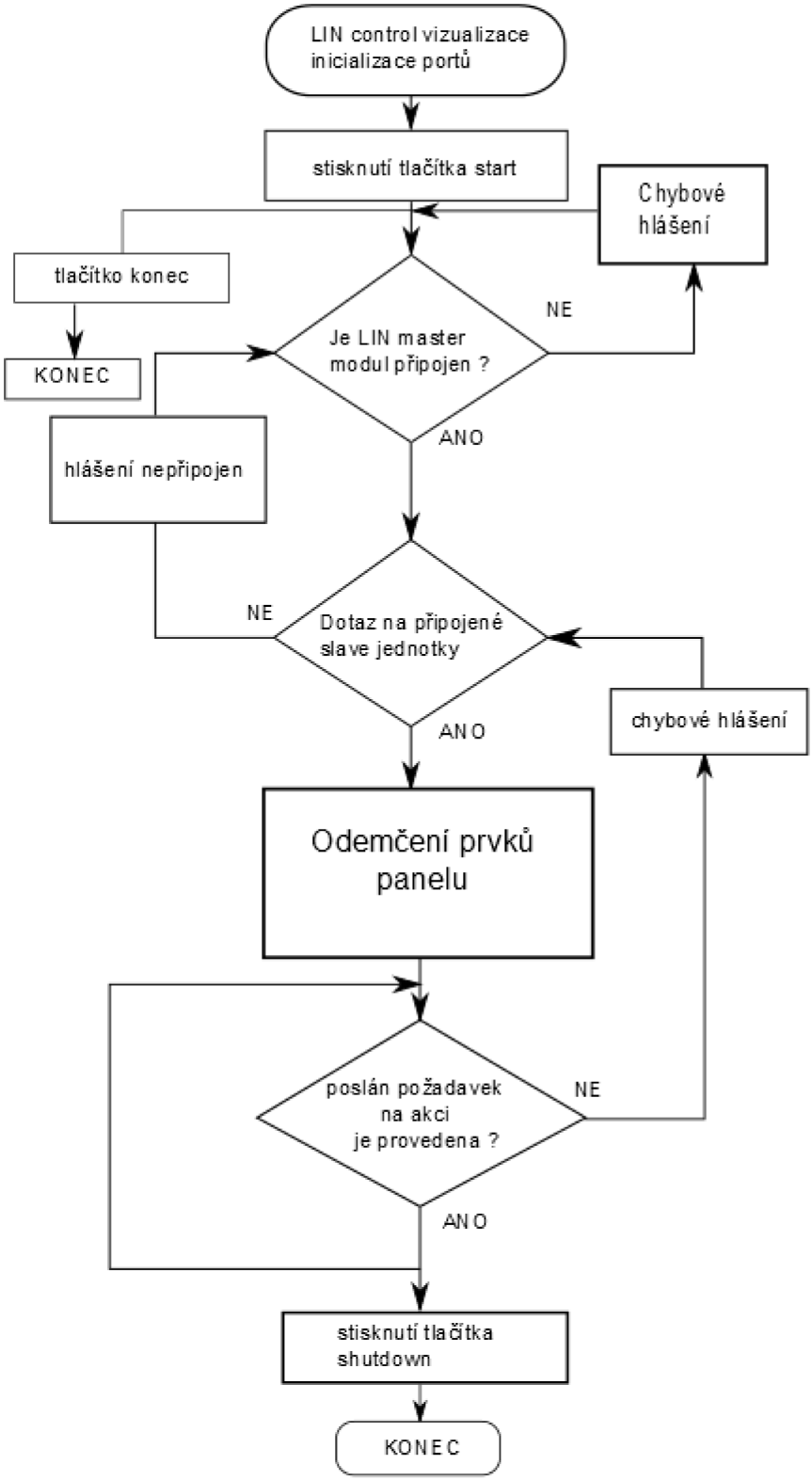
3.4 Řídící vizualizace

Samotné řízení je realizováno ve vizualizaci jež běží v PC pro kontrolu motorů nebo zobrazování dat na LCD displayi. Když je program spuštěn snaží se komunikovat přes USB se samotným master modulem není-li k dispozici dojde k vyvolání chybového hlášení že není dostupný master model, je-li dostupný pracuje jak popisu (vývojový diagram lze vidět na obrázku č.9) pro samotný komunikační port lze nastavit jen komunikační rychlost. Samotné data proudí jako Master data čímž je zachována priorita řídícího zásahu protože SCI2 funguje v režimu AS1.

Vývoj aplikace byl proveden v jazyce C#, jež patří do skupiny OOP jazyků to znamená, že není nutno samotné objekty programově definovat, jsou již vytvořeny a jsou jen parametrovány. pro její realizaci bylo užito prostředí Visual studio 2010 Ultimate edition jež Microsoft poskytuje studentům jež mají zapsány předměty související s informatikou a pokud není užita ke komerčním účelům je poskytnuta zcela zdarma. Díky uživatelskému prostředí je vývoj samotné aplikace jednoduchou záležitostí jelikož jsme schopni velmi jednoduše nadefinovat uživatelské prostředí(viz. Obr.9).

Podoba samotné vizualizace byla navržena pro jednoduché prostředí pro snadnou manipulaci, kde posuvem slideru změníte rychlost nebo změníte směr v případě zaslání zprávy do LCD zadáte text do rich textboxu kde je limitován 32 znaky jelikož se výstup promítne do 2*16 řádkového displaye. V případě chodu motoru je jsou užity šipky pro orientaci směru kde a tlačítkem stop můžeme zastavit chod modulu a tlačítkem shutdown jej kompletně vypnout (samotnou strukturu programu lze vidět v rámci obrázku č.10).

Obr. 9: Vizualizace



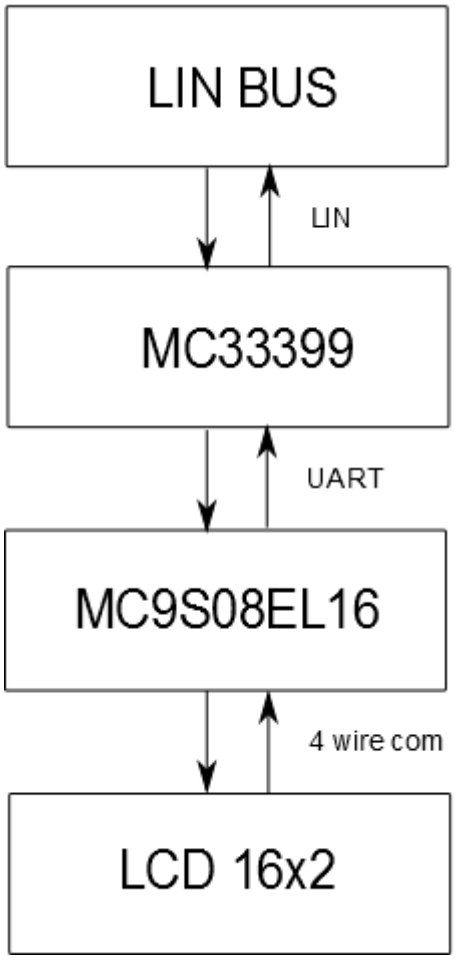
Obr. 10: Struktura vizualizace

4. Slave jednotka – LCD

Základním účelem tohoto modulu je zobrazování informace o stavu zařízení jež jsou k linu připojena nebo zobrazovat hlášení o chybách jež vznikly v rámci komunikace pro samotné sběrnici, pro tuto aplikaci jsem si zvolil procesor MC9S08EL16CTJ jež je výbornou variantou pro realizaci slave zařízení jež je primární účel této řady ,ale jelikož nemá vyvedena PWM tak byl užit pouze pro ovládání LCD displaye.

Důležité čipy LCD modulu:

- MC33399 - LIN driver
- MC9S08EL16CTJ – mikroprocesor slave jednotky
- LCD 16x02 - POWERTIP PC1602F



Obr. 11: Struktura LCD SLAVE modulu

4.1 Popis zapojení

Obdobně jako u jednotky master je i u slave jednotky řešeno napájení. Vstup regulátoru napětí je opět obvod 7805, je připojený na 12V z LIN sběrnice. Výstup obvod 7805 je 5 V pro napájení MCU a LCD displaye. Je zde také přítomna Shottkyho dioda, která chrání obvod před přepolovaním a zelená LED dioda, která signalizuje připojení jednotky k napájení. Před LED diodou je ochranný rezistor 10 *kΩ*. Mikroprocesor Freescale MC9S08EL16CTJ je napájený 5 V z regulátoru 7805. O rychlost (hodiny) MCU se stará externí zdroj hodinového taktu, jež je realizován krystalem o hodnotě 16 *MHz* (připojen na piny *EXTAL a XTAL).* I slave jednotku je potřeba naprogramovat případně ladit, tzn. i zde je přítomný konektor BDM. Na piny Portu A je připojen LCD display, LIN driver je také stejný, MC33399 od firmy Freescale. Napájený je přímo z LINu 12 *V.* Zde je změna oproti jednotce master v tom, že jednotky slave nemají pull-up rezistor, datový pin LIN je připojený přímo ke sběrnici. Piny RxD a TxD jsou připojené k MCU na piny SCI. Ostatní piny LIN driveru jsou připojeny stejně jako u zařízení master, a to následovně: vstup EN = log. 1, vstup WAKE = log. 0 a vystup INH není připojeny. Na pinu (PTA3) je připojeny pouze jedním datový vodičem sběrnice 1-Wire. Dále je zde připojen LCD display který jsem užil je POWERTIP PC1602F s řadičem, je to LCD s dvěma řádky a 16-ti znaky na jeden řádek (2x16) je připojen na port A, jelikož se bude používat LCD jen jako zobrazovač nebudeme z něj nic vyčítat, pin R/W jsem spojil se zemí. Na pevno je také připojeno podsvícení displaye. Pin V0 je určen pro je pro regulaci jasu a je spojen s potenciometrem . Datové piny jsou připojeny 4 vodičovým způsobem proto musí být dolní polovina spojena se zemí jelikož se nepoužívají. Horní polovina pinů je připojena na port A na MCU. Dva zbývající řídící piny displaye jsou taky připojené k MCU, pin ENABLE (E) a Register select. Piny jsou vyvedené na malý konektorek 2x3 včetně napájeni (+5 *V* a GND).BDM konektor je velmi důležité, je to rozhraní, které se používá pro programování laděni (debug) programu (firmwaru) do MCU. Zapojeni BDM konektoru (u popisu jednotky master). Z MCU je pro programování/laděni vyvedeny pouze jeden signál a to BKGD pin druhý. Schéma a DPS je umístěna v příloze B.[1][4][5][11]

4.2.1 Mikrokontrolér MC9S08EL16

Mikrokontrolér který je užit ve Slave jednotce je také 8-bitový MCU s jádrem HCS08. Tento MCU je přímo určen pro slave jednotky, protože má periferii SLIC, což znamená že má implementován LIN slave řadič, tedy plné hardwarové řešení LINu. [4][5][8]

Základní vlastnosti:[4]

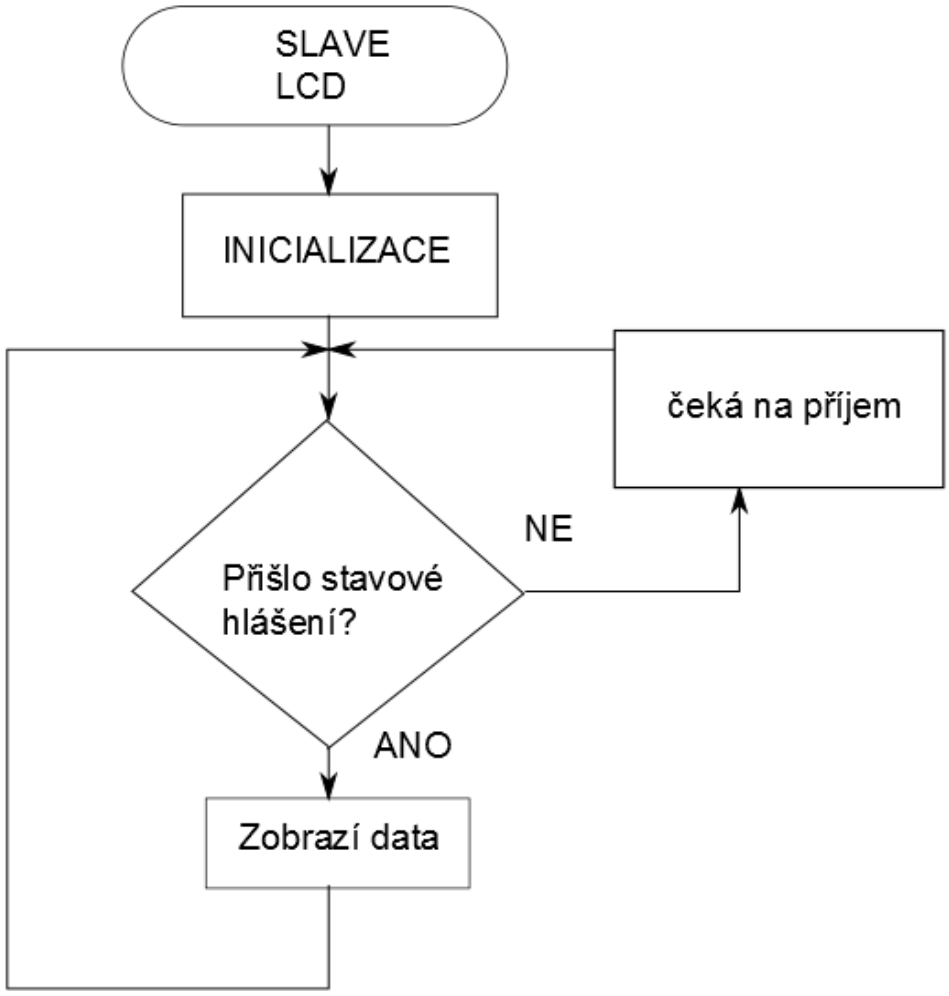
- Napájeni od 2.7 do 5.5 V.
- jádro HCS08 může běžet až na 40 MHz a vnitřní sběrnice až na 20 MHz.
- 16 kB flash.
- 512 B RAM.
- 256 B EEPROM.
- Možnost využiti vnitřního (Internal Clock Source) nebo externiho hodinového taktu.
- Má 16 I/O pinů.
- Má jednovodičové rozhraní pro programování a laděni (BDM).
- **Systémová ochrana:**
- Computer Operating Properly (COP) Watchdog.
- Low Voltage Detect.
- Illegal address detection.

- Flash and EEPROM block protect.

- **obsahuje tyto periferie:**
- 1x SLIC, 1x I2C, 1x SCI a 1x SPI.
- SLIC podporuje LIN ve verzi 2.0.
- obsahuje jeden 2-kanálový, 16-bitový časovač.
- Obsahuje jeden 10-kanálový, 12-bitový A/D převodník.
- Má dva analogové komparátory a RTC (Real-time counter).

4.2 Programování LCD SLAVE modulu

Tělo samotného programu jakožto i řídící smyčka for(;;) je umístěna v main.h, v případě slave modulů je potřeba inicializovat SCI do režimu tkzv. AS_SCI což znamená že již v rámci inicializace procesoru určíme tuto komunikaci jako podřízenou a v procesor expertu vytvoříme informace o užívaném LIN driveru, což je následně jednodušší pro samotnou práci na komunikaci. Dále na portu A



Obr. 12: Vývojový diagram programu LCD Slave modulu

nastavíme kontrolní piny pro dodání informací na LCD display 16x2.

Po ukončení inicializace vstoupíme do samotné smyčky programu *for ();* V samotné smyčce jednotka užívá header LIN zprávy a očekává identifikační volání na požadavky provedení úkonu. V rámci komunikace je této jednotce přiřazeno Slave ID 1 pro lepší rozlišení jednotek kde pro inicializované identifikace si master uzamkne ID pro tuto jednotku v následujících případech pro SS

motor toto Slave ID bude 2 a pro krokový motor 3 a bude umístěno v obsluze rutin jak v Master jednotce tak ve všech jednotkách Slave.

Header soubory užité v programu:

- LinMC33399.h
- LCD.h
- cekej.h
- main.h

LinMC33399.h – jedná se o modifikovanou variantu knihovny MC33399.h jež obsahuje jen rutiny pro podřízený chod

cekej.h – definování časového zpoždění

LCD.h – obsahuje definici obsluhovaného portu A odkud se odesílá rámec zprávy pro zobrazení

*void **LCDZAPISDATA**(unsigned char **data**)* – výpis znaku na samotný LCD display.

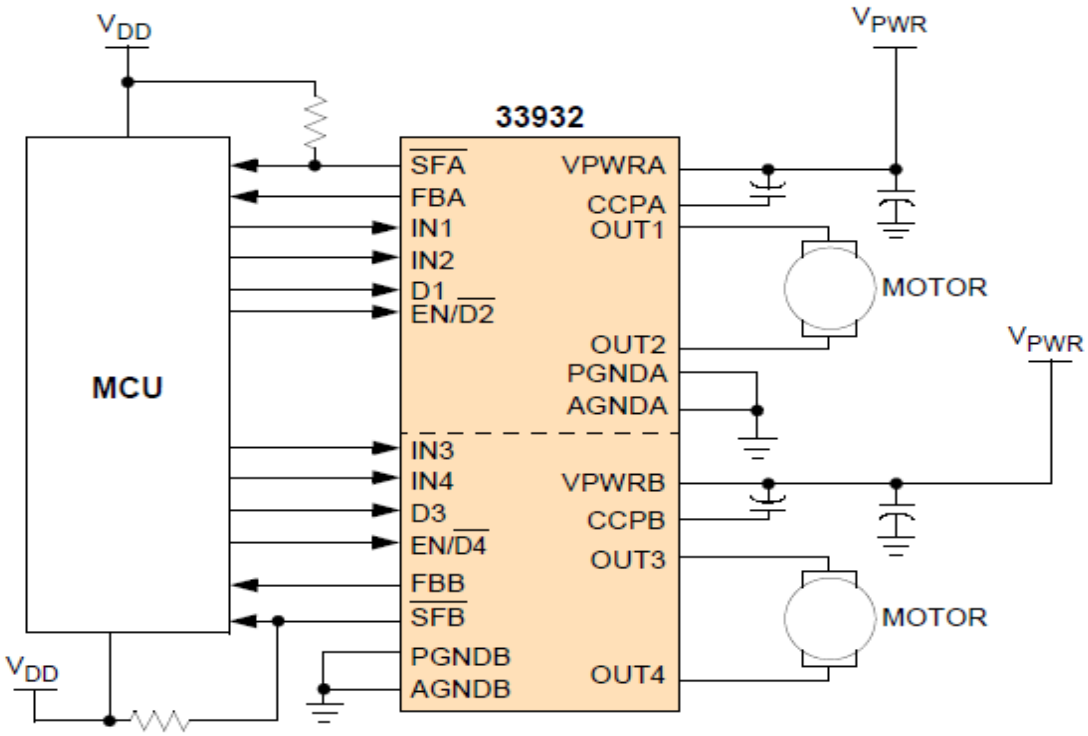
*void **LCDINIT**(void)* – inicializace samotného LCD displaye resp. Natavení portu A pro odesílání rámců pro zápis na display

*void **LCDSMAZ**(void)* – funkce pro smazání celého LCD displaye

*void **LCDPOZICE**(unsiged char **radek**, unsiged char **sloupec**)* – funkce pro osazení znaků na display

5. Slave jednotka – SS motorek

Pro řízení SS motorku padla volba na užití robustního driveru MC33932 jež se často užívá pro řízení serv a modelářských SS motorků, jelikož byl již použit při soutěži freescale race chalange. Hlavní výhodou je to že pro připojení motorku postačuje kromě MCU velmi málo dalších komponent a v případě nahrazení současného slave procesoru jiným procesorem lze řídit najednou až 2 motorky (viz. Obr. 14).



Obr. 13: Užití driveru MC33932 [8]

Navíc tento driver je velice robustní a vydrží i značné proudy a zahřívání jelikož je připraven i na případné reverze směru otáčené motorku a tím pádem na zpětné proudy jdoucí z motorku do daného driveru, což vedlo k přizpůsobení desek zvýšením plochy a přidáním prokovek.

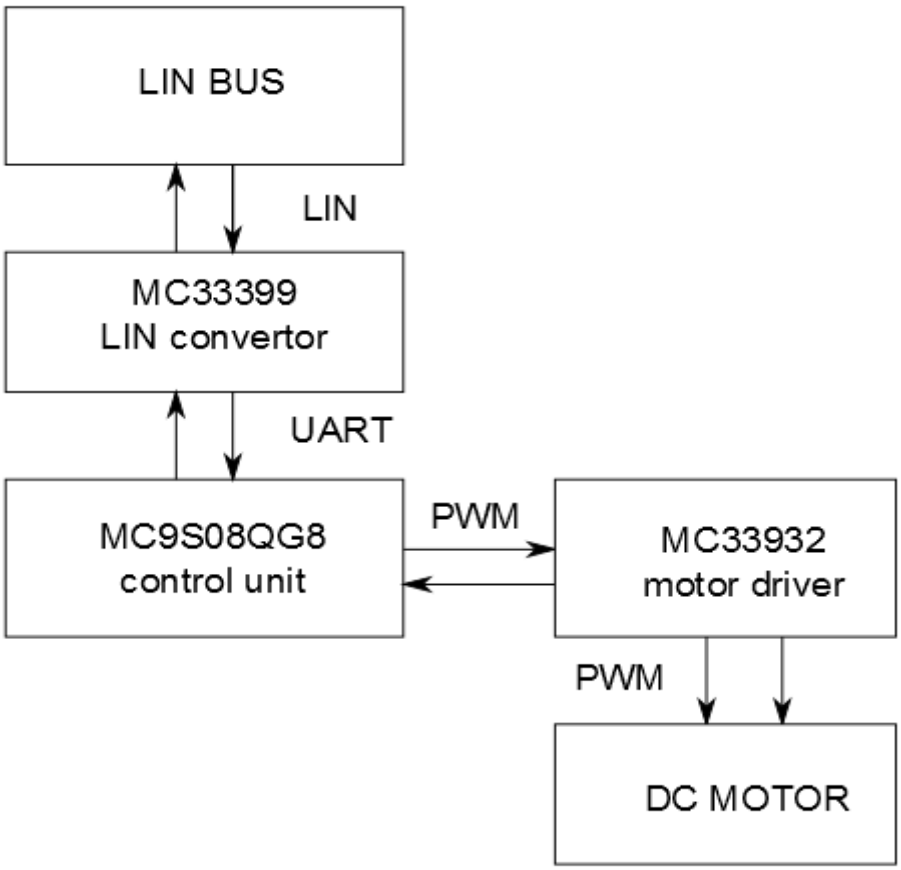
Pro řízení motorových modulů jsem užil procesor MC9S08QG8 jež je přímo určen pro řízení motorů.

Důležité čipy ss motorového modulu:

- MC33661 - LIN driver PIN to PIN kompatibilní k MC33399 ,ale je schopný pracovat na 3.3V
- MC9S08QG8 – mikroprocesor slave jednotky
- LT33 – stabilizátor napájecího napětí
- MC33932 - driver pro SS motor

5.1 Struktura SLAVE modulu pro Stejnosměrný motor

Mikrokontrolér který je užit ve Slave jednotce je také 8-bitový MCU s jádrem HCS08. Tento MCU je přímo určen pro řízení motorků jedná se o low power high performance čip jehož vlastní CPU má takt 20 MHz, protože má v sobě implementován SCI jednoduše jej použiji pro vytvoření LIN komunikace skrze LIN řadič.

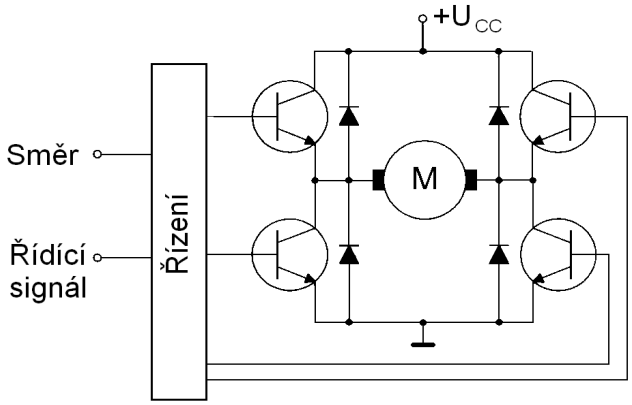


Obr. 14: Struktura SS Motor SLAVE modulu

5.1.1 Popis zapojení

Obdobně jako u jednotky master je i u slave jednotky řešeno napájení. Vstup regulátoru napětí je obvod LT33, je připojený na 12V z LIN sběrnice. Výstup obvod LT33 je 3.3 V pro napájení MCU pro motor je napájení externí přímo na driver. Je zde také přítomna Shottkyho dioda, která chrání obvod před přepolovaním a zelená LED dioda, která signalizuje připojení jednotky k napájení. Před LED diodou je ochranný rezistor 10 kΩ. Mikroprocesor Freescale MC9S08QG8 je napájený 3 V z regulátoru LT33. O rychlost (hodiny) MCU se stará externí zdroj hodin realizovaný krystalem o hodnotě 16 MHz (*připojen na* piny EXTAL a XTAL). I slave jednotku je potřeba naprogramovat případně ladit, tzn. i zde je přítomný konektor BDM. Na piny TMPCH je připojeny na Driver, LIN driver je MC33661 od firmy Freescale. Napájený je přímo z LINu 12 V. Zde je změna oproti jednotce master v tom, že jednotky slave nemají pull-up rezistor, datový pin LIN je připojený přímo ke sběrnici. Piny RxD a TxD jsou připojené k MCU na periferii SCI. Ostatní piny LIN driveru jsou připojeny

stejně jako u zařízení master, a to následovně: vstup EN = log. 1, vstup WAKE = log. 0 a vystup INH není připojeny. Na 17 pinu (PTA3) je připojeny pouze jedním datový vodičem sběrnice 1-Wire. Dále je zde připojen Driver MC33932 (vnitřní zapojení a jeho užití je naznačeno zjednodušeně naznačeno v obr. č.15) a je aktivován jen můstek A pro pohon SS motoru dále je zde zhuštěno filtrování proti zemi jelikož při reverzaci dochází k velkým zpětným proudům. Pinem MCU PTA0 a PTB5 přivedeme řídící PWM signál můstek na piny IN 1 a 2 kde dojde k samotnému ovládání výstupu OUT 1 a 2 pro samotný chod je důležité taky můstek aktivovat náběžnou hranou přes PIN D1 a naopak poslat negovaný signál na pin EN2/D2 neg. Samotný můstek je napájen přímo z externího zdroje (autobaterie). Schéma a DPS je vloženo v příloze C.[10][8][11]



Obr. 15: Struktura driveru pro řízení motoru [9]

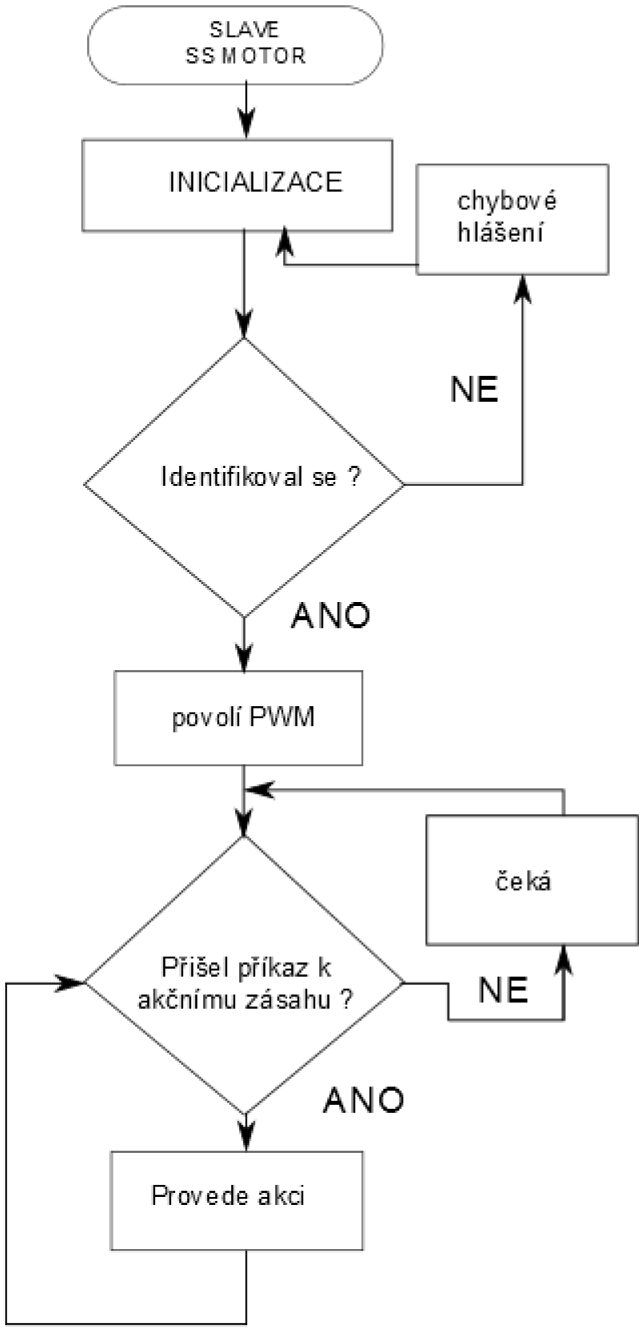
5.1 Programování Slave modulu

Tělo samotného programu jakožto i řídící smyčka for(;;) je umístěna v main.h, v případě slave modulů je potřeba inicializovat SCI do režimu tkzv. AS_SCI což znamená že již v rámci inicializace procesoru určíme tuto komunikaci jako podřízenou a v procesor expertu vytvoříme informace o užívaném LIN driveru, což je následně jednodušší pro samotnou práci na komunikaci. Dále musíme nastavit zabudovaný časovač do PWM režimu pro řízení samotného motoru, dále je také nutno využít režimy samotného PWM dle užití.

Po ukončení inicializace vstoupíme do samotné smyčky programu *for (;;);* V samotné smyčce jednotka užívá header LIN zprávy a očekává identifikační volání na požadavky provedení úkonu.

Hlavičkové soubory užité v programu:

- *LinMC33399.h*
- *PWM.h*
- *cekej.h*
- *main.h*



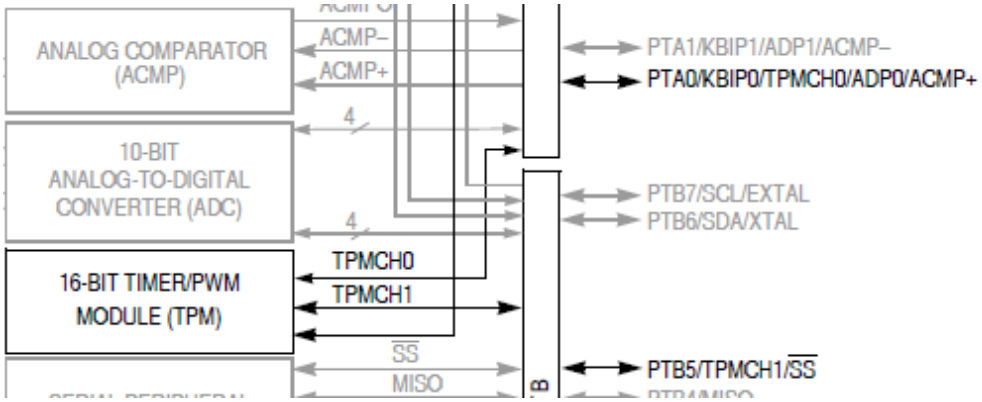
Obr. 16: Vývojový diagram Slave modulu pro řízení SS motoru

main.h - zde je umístěn samotný inicializační program a obsluha přijatých řídících rámců.
LinMC33399.h – stejně jako v předchozím případě je to upravená verze LIN pro SLAVE chod
PWM.h – zde jsou umístěny předefinované metody pro ovládání směru a rychlosti kde jednoduše předefinujeme buzení PWM na daných pinech v závislosti na logické tabulce daného driveru.

void ***DCFORWARD*** *(long* ***rychlost)*** – povolí první PWM pro chod dopředu
void ***DCREVERSE****(long* ***rychlost)*** – povolí druhé PWM pro reverzní chod

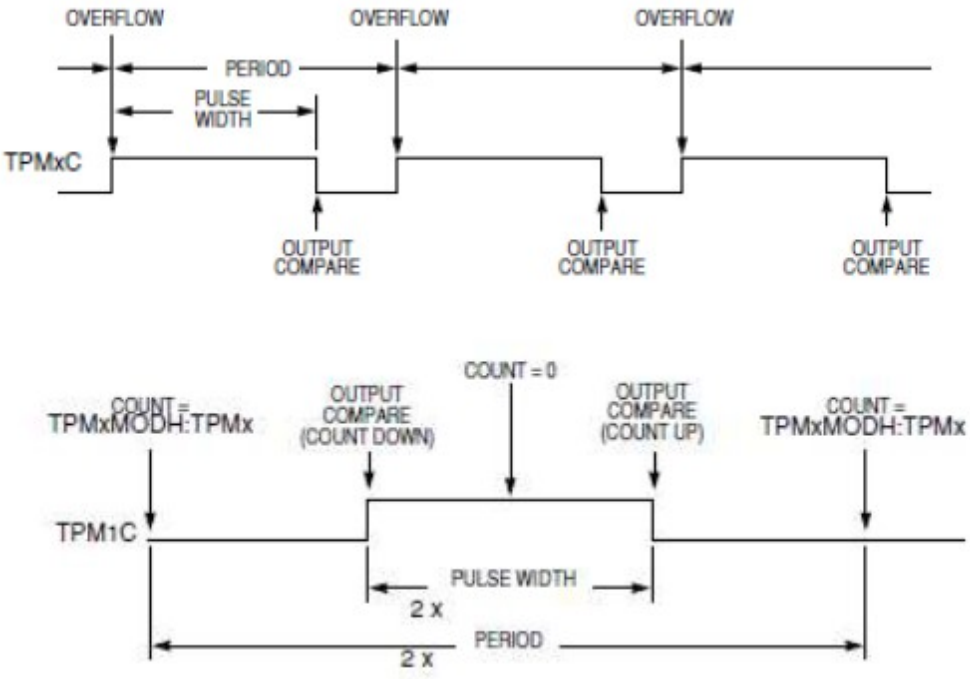
5.2.1 PWM řízení

Pro řízení mé aplikace jsem využil časovače v procesoru kde má pro každý kanál přiřazen jeden 8mi bitový registr(viz. Obr. č.17), ale nejdříve přes data direction registr jej nutno povolit pro samotný Port A a B.[11]



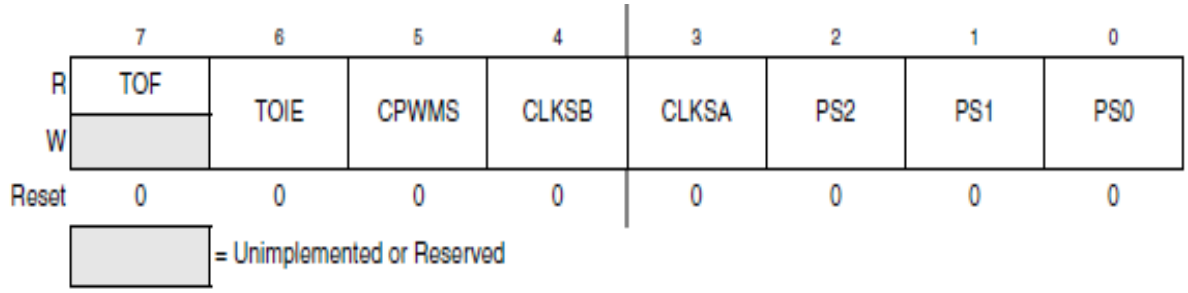
Obr. 17: Výstup časovače/PWM modulu [11]

MCU chod. Samotný průběh PWM je definován po naplnění samotného registru a po přetečení nastává nový pulz. Jehož průběh a možnosti lze vyčíst z následujícího obrázku (viz. Obr. č.18). V tomto př9padě nevyužíváme pro PWM celých 16 *Mhz* ,ale jen 12 *MHz* [11]



Obr. 18: PWM režimy [11]

hlavním činitelem v užívání PWM je pro tuto činnost je TPMSC kde je kontrolováno nastavení samotného časovače. Kdo TOF kontroluje přetečení registru. [11]



Obr. 19: Rozložení TPMSC registru [11]

TOF – timer overflow flag který signalizuje přetečení samotného registru jež je kontrolován čítáním

TPM = 0 čítač ještě nedosáhl stavu přetečení

TPM = 1 čítač dosáhl stavu přetečení [11]

TIOE - Timer Overflow Interrupt Enable

umožňuje povolit vyvolání přerušení při přetečení registru TOF v případě že:

TOF = 0 je vyvolání přerušení potlačeno a musíme užít programového dotazování

TOF = 1 přerušení jsou povolena a je možno reagovat ne přetečení registru.[11]

CPWM - Center-Aligned PWM Select tímto registrem nastavujeme operační mód PWM

CPWM = 0 v tomto režimu PWM funguje podle náběžný hran a podle toho je možno i realizovat zde užívané PWM.. Toto nastavení funguje pro oba kanály.

CPWM = 1 všechny kanály jsou nastaveny na středem zarovnané PWM.[11]

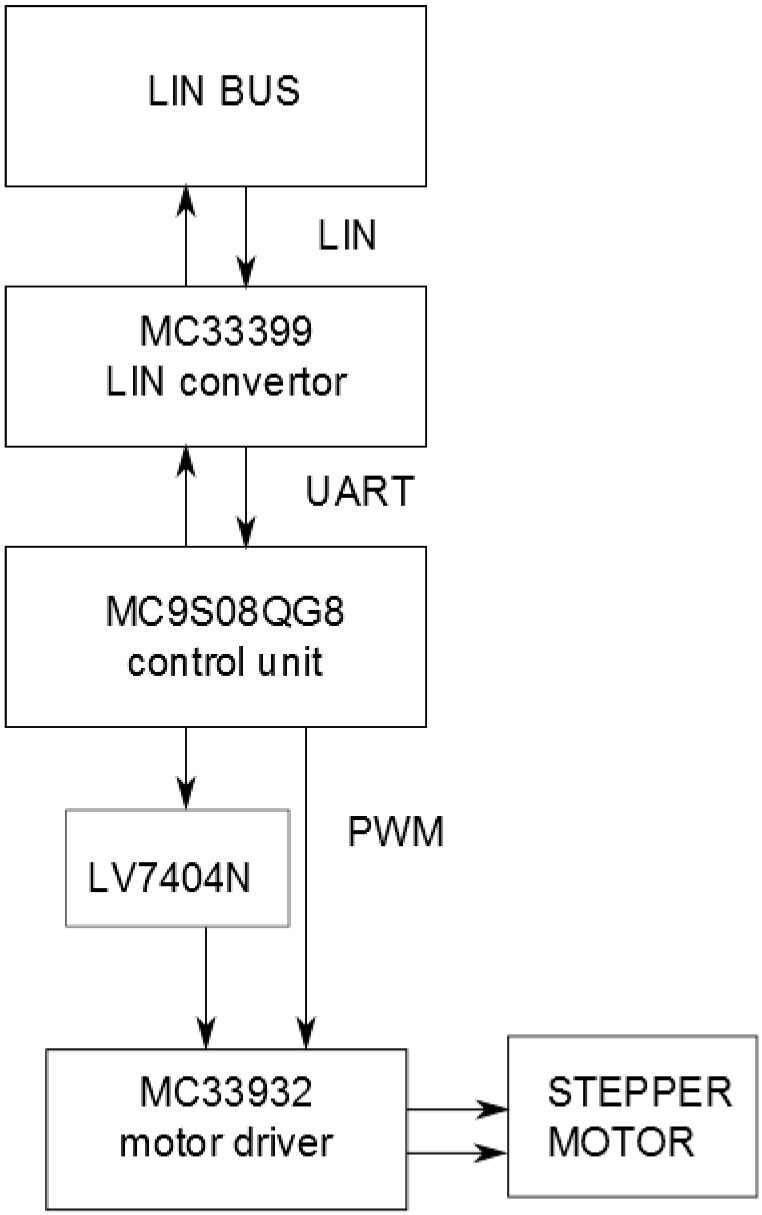
CLKS [A,B]Clock Source Select je to dvoubitové pole které se užívá pro určení zdroje časového signálu pro oba kanály časovače a nebo pro selekci před vzorkovačů, časování jak vnitřního a vnějšího krystalu je synchronizováno samotným MCU.[11]

6 Řídící Slave jednotka pro krokový motor

Pro řízení krokového motorku byl užít robustní driver MC33932 Hlavní výhodou je to že pro připojení motorku postačuje kromě MCU velmi málo dalších komponent a v případě nahrazení současného slave procesoru na rozdíl od předchozího případu ale užívám oba můstky pro řízení krokového motoru.

Důležité čipy ss motorového modulu:

- MC33661 - LIN driver PIN to PIN kompatibilní k MC33399 ale je schopný pracovat na 3.3V
- MC9S08QG8 – mikroprocesor slave jednotky
- MC33932 – driver pro řízení motorů
- SN74AC04N – invertor pro další kanály dále označován jako LV7404N



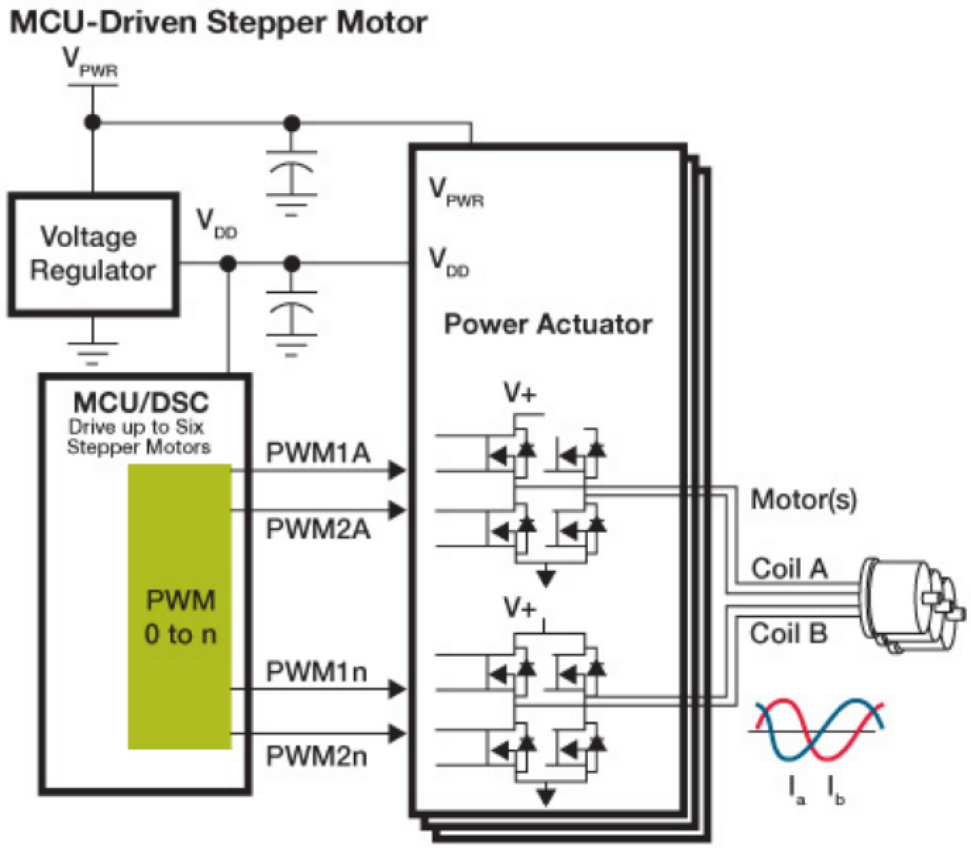
Obr. 20: Blokové schéma Slave modulu pro řízení krokového motoru

navíc tento driver je velice robustní a vydrží i značné proudy a zahřívání jelikož je připraven i na případné reverze směru otáčené motorku a tím pádem na zpětné proudy jdoucí z motorku do daného driveru, což vedlo k přizpůsobení desek zvýšením plochy a přidáním prokovů.

Pro řízení motorových modulů jsem užil procesor MC9S08QG8 jež je přímo určen pro řízení motorů.

6.1 popis zapojení

Obdobně jako u jednotky master je i u slave jednotky řešeno napájení. Vstup regulátoru napětí je opět obvod LF33, je připojený na 12V z LIN sběrnice. Výstup obvod LF33 je 3.3 V pro napájení MCU pro motor je napájení externí přímo na driver. Je zde také přítomna Shottkyho dioda, která chrání obvod před přepólováním a zelená LED dioda, která signalizuje připojení jednotky k napájení. Před LED diodou je ochranný rezistor 10 kΩ. Mikroprocesor Freescale MC9S08QG8 je napájený 3 V z regulátoru LF33 O rychlost (hodiny) MCU se stará externí zdroj hodin se stará externí zdroj hodin realizovaný krystalem o hodnotě 16 MHz (*připojen na* piny EXTAL a XTAL). I slave jednotku je potřeba naprogramovat případně ladit, tzn. i zde je přítomný konektor BDM. Na piny TMPCH je připojen driver pro motor a invertor, LIN driver je také stejný, MC33661 od firmy Freescale. Napájený je přímo z LINu 12 V. Zde je změna oproti jednotce master v tom, že jednotky slave nemají pull-up rezistor, datový pin LIN je připojený přímo ke sběrnici.



Obr. 21: Blokové schéma řízení Brokového motoru [15]

Ostatní piny LIN driveru jsou připojeny stejně jako u zařízení master, a to následovně: vstup EN = log. 1, vstup WAKE = log. 0 a vystup INH není připojeny. Na 17 pinu (PTA3) je připojeny pouze jedním datový vodičem sběrnice 1-Wire. Dále je zde připojen Driver MC33932 a jsou aktivovány

můstky A a B pro pohon krokového motoru dále je zde zhuštěno filtrování proti zemi jelikož při reverzaci dochází k velkým zpětným proudům. Pinem MCU PTA0 a PTB5 přivedeme řídící PWM signál můstek na piny IN 1 a 3 řízenou PTA0 a druhou dvojici IN 2 a 4 řízenou PTB5 bude přiváděn signál pro reverzaci směru jelikož máme k dispozici jen 2 kanály PWM bude jednoduší užit LV7404N pro inverzi PWM signálu. Jelikož řízení krokového motoru je v podstatě pulzní řízení 2 motorů jako v tomto případě kde je užit 4 pólový krokový motor. Je užito obou můstků pro realizaci řízení samotných motorů první cívka motoru je řízena výstupy OUT 1 a 2 a druhá cívka OUT 3 a 4. Schéma zapojení,DPS a logická tabulka driveru je uvedena v příloze D[6][8][11][15]

6.2 Programování Slave modulu

Tělo samotného programu jakožto i řídící smyčka for(;;) je umístěna v main.h, v případě slave modulů je potřeba inicializovat SCI do režimu tkzv. AS_SCI což znamená že již v rámci inicializace procesoru určíme tuto komunikaci jako podřízenou a v procesor expertu vytvoříme informace o užívaném LIN driveru, což je následně jednodušší pro samotnou práci na komunikaci. Dále musíme nastavit zabudovaný časovač do PWM režimu pro řízení samotného motoru, dále je také nutno využít režimy samotného PWM dle užití kde musím brát v potaz užití invertoru pro druhý můstek.

Po ukončení inicializace vstoupíme do samotné smyčky programu *for (;;)* V samotné smyčce jednotka užívá header LIN zprávy a očekává identifikační volání na požadavky provedení úkonu.

Hlavičkové soubory užité v programu:

- *LinMC33399.h*
- *krokac.h*
- *cekej.h*
- main.h

main.h – zde je uloženo tělo programu s inicializací a pro obsluhu událostí jako identifikování,přetečení bufferu atd..

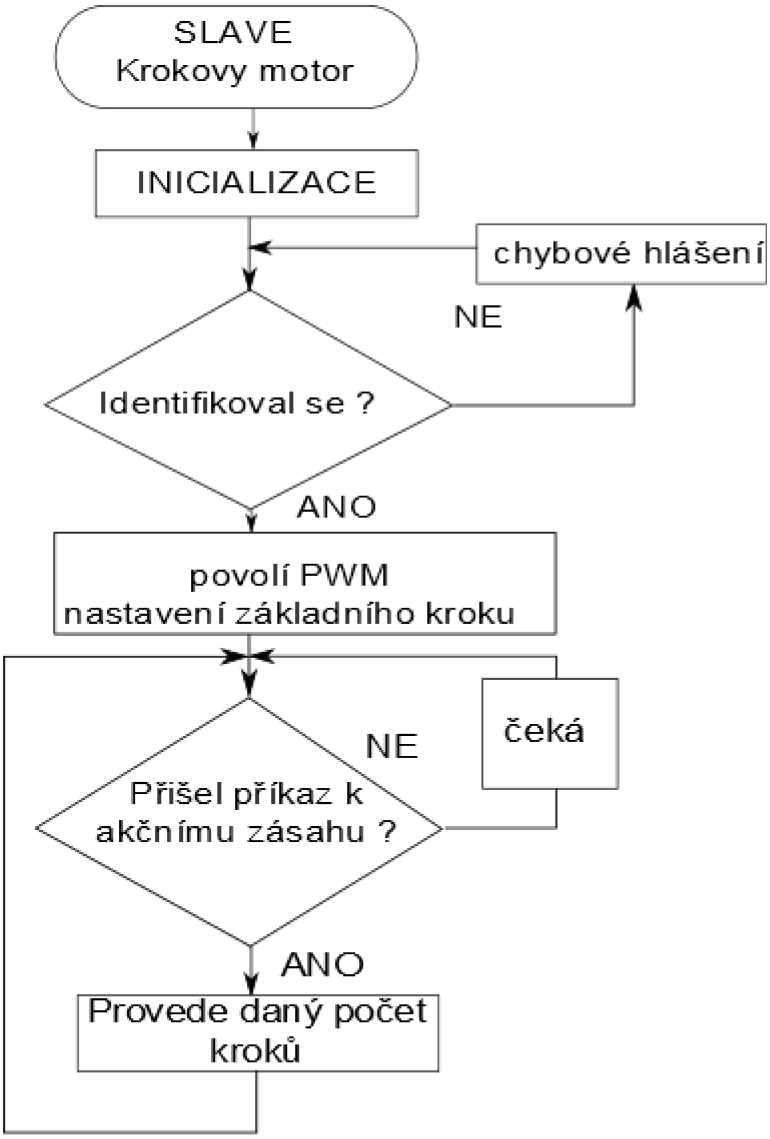
cekej.h – zde je nastaveno spoždení stejně jako v předchozích případech.

krokac.c - je to funkce pro obsluhu PWM pro nastavení chod motorů

void ***STEPFORWARD*** *(long* ***rychlost)*** – povolí první mód PWM pro chod dopředu

void ***STEPREVERSE****(long* ***rychlost)*** – povolí druhý mód PWM pro reverzní chod jež jdou v logic low módu aby invertor obrátil orientaci na high u daného PWM na zpětný chod (viz. Logická tabulka pro driver v příloze D)

LinMC33399.c – rutiny pro obsluhování LINu pro přijetí zprávy a předání akce dle požadavku



Obr. 22: Vývojový diagram slave modulu krokové jednotky

7. Zhodnocení výsledků

Cílem této práce bylo seznámit se s užitím a provozováním LIN sběrnice pro měření a řízení, navržení Hardwaru pro komunikaci s PC z čehož plyne i nutnost vytvoření uživatelského prostředí, poté návrh dalších dvou modulů pro stejnosměrný a krokový motor, jež jsou konstrukčně nejnáročnější a nakonec vytvoření posledního hardwarového modulu s LCD displayem pro zobrazování dat. Úvodní částí je analýza LIN sběrnice , nezbytná část pro pochopení daného problému komunikace a pro programování komunikace je klíčová.

Druhým krokem je návrh master modulu pro komunikaci s PC, zde již při samotné konstrukci a výběru součástek je nutno si uvědomit ,že dochází k přechodu v komunikaci nejen UART/LIN ,ale také pro přechod mezi MCU (zde zastupován periferií SCI) a portem USB v počítači. Na samotné desce je nezbytný LIN driver jež převádí napěťové úrovně LIN sběrnice na úroveň UARTu. Samotný driver pro LIN (MC33399 a MC33661) od firmy Freescale je problematické získat v menším množství. Jelikož se jedná o masově užívaný díl pro automotive průmysl není možnost získání vzorků a minimální odběr činí 10000 kusů. Malé množství těchto driverů bylo věnováno firmou Freescale. Neméně důležitý je vhodně zvolený mikrokontrolér ,který je schopen plnit kladené požadavky a měl veškeré požadované periferie. Samotné oživování probíhalo najednou desku jsem před osazování otestoval na propojené cesty nikde nebyla viditelná známka problému na desce došlo k osazení a průběžné kontrole studených spojů deska byla oživena na první pokus.

Druhým krokem bylo zprovoznění FTDI pro komunikaci s PC, kde po úspěšné instalaci driveru pro Windows 7 a připojení zařízení se po několika pokusech samotné zařízení zobrazilo v PC. Po té jsem vytvořil vizualizaci pro uživatele se zjednodušenými příkazy pro otestování funkčnosti komunikace mezi PC a MCU, po ověření této komunikace. Jsem přešel k nadefinování softwaru pro master jednotku v rámci komunikace po sběrnici LIN, kde jsem nadefinoval rutiny pro očekávané funkce. V následujícím kroku bylo nutno vytvořit jedno Slave zařízení pro zprovoznění LIN komunikace. Prvním zhotoveným Slave modulem je LCD na němž lze snadno demonstrovat funkční komunikaci master/slave posláním znaku k zobrazení. Samotné oživení proběhlo již postupně, to znamená ,že v průběhu osazování byla provedena kontrola funkčnosti. Po oživení bylo nutno doladit jas na rozumnou míru potenciometrem při prvním testu samotná aplikace odebírala skoro půl ampéru při neregulovaném jasu. Pro komunikaci na tomto modulu je nutno vytvořit knihovnu s omezeným počtem instrukcí pro obsluhu slave jednotky a umístí se do složky source souborů. Pro obsluhu LCD displaye pro je nutno nadefinovat funkci pro poziční umisťování znaků. Během programování je velmi hojně užito aplikace procesor expert pro vytváření inicializačních souborů. Jelikož jsme schopni přímo definovat master/slave prioritu SCI, proto přímo funkce AS navážeme na knihovnu obsluhující LIN sběrnici, pro chybové stavy jsou vytvořeny obsluhy událostí.

V dalším kroku proběhla realizace modulu pro řízení stejnosměrného motoru. Zde bylo nutno nastudovat problematiku stejnosměrných motorů ve smyslu jejich provozu a řízení. Dále bylo nutno nastudovat ovládání stejnosměrných motorů skrze PWM a aplikování driveru s H můstkem. Samotný vývoj hardwaru byl problematický z důvodu zohlednění tepelných vlastností driveru v provozu a dvou oddělených napájení jedno pro samotný procesor a druhé pro samotný driver. Tento driver napájí a řídí motor v případě přetížení (během reverzace) by mělo dojít jen k poškození driveru. Během samotného oživování byl problém z hlediska leptání jelikož rozvor nožiček u driveru byly na mezi proveditelnosti v rámci daného vývojového pracoviště, tento problém se opakoval i u následujícího modulu. Po oživení byly nahrány připravené konstrukce kódu pro PWM a obsluhu portů po dvou testech byl stejnosměrný motor zprovozněn. Poté bylo nutno zprovoznit LIN komunikaci, zde je opět užita variace souboru LinMC33399.c pro předávání kontrolních rámců mezi modulem a master jednotkou.

V rámci samotného programu bylo také nutno zadefinovat vynucené zpoždění během změny směru pro ochranu samotného driveru . Po několika pokusech se povedlo zprovoznit komunikaci v rámci LIN sběrnice předáním informace o rychlosti a rozjetím samotného motorku. Po zprovoznění druhého modulu bylo nutno zadefinovat samotné slave adresy, aby v případě nenadálého odpojení jednoho z modulů nedošlo k poslání příkazu na špatný modul.

Při návrhu posledního z modulů bylo nutno pečlivě nastudovat odlišnosti chování krokového motoru vůči samotnému driveru jež je stejný jako v předchozím případě, ale bylo nutno užít oba můstky. Jelikož dodaný motor je bipolární krokový motor a je žádoucí zpětných chod byl do konstrukce přidán invertor LV7404, jež umožní se dvěma PWM kontrolovat oba můstky i. Během programování bylo nutno nejprve odzkoušet možnosti inverze signálu. Při osazování desky se objevil problém v podobě prodlení dodávky druhého driveru, jelikož druhý z driverů byl dodán nefunkční.

V rámci oživení desky se objevily problémy s viditelně proleptanou cestou jež proleptána nebyla. Jiné problémy v průběhu oživování zaznamenány nebyly. V posledním kroku jež je programováním tohoto modulu bylo problematické vyladit PWM pro bezproblémový chod na samotném můstku. Na straně rozcházení komunikace nastal zásadní problém. Samotný výstup z LIN driveru MC33661 neodpovídal chování tohoto driveru v předchozí aplikaci tento problém byl vyřešen až vyměněním driveru. Po tomto kroku bylo nutno navázat deklarované funkce v slave modulech do master modulu a vytvořit reakční události v závislosti na posílaných příkazech a dolazení propojeni skrze FTDI jež nebylo úplně ideální a v rámci samotného opětovného uvedení do provozu, proto by bylo pro příště lepší užít přímo USB driver od firmy freescale. Samotné vizualizační rozhraní vyžaduje ještě čas pro dolazení funkčnosti v rámci samotné struktury a určení událostí jež by měli být vyhodnoceny.

8 Použitá literatura

[1] OTEVŘEL,V. *Využití LIN sběrnice v řidicích systémech*. Ostrava, 2010. Bakalářská práce na Fakultě elektrotechniky a informatiky VŠB-Technické univerzitě Ostrava na katedře měřicí a řídicí techniky. Vedoucí práce Zdeněk Slanina.

[2] Švimberský Z., *LIN – Local Interconnect Network*, Bakálařská prace, Praha, 2007 URL: <http://support.dce.felk.cvut.cz/mediawiki/images/7/76/Bp_2007_svimbersky_zdenek.pdf >

[3] Tým autorů Freescale Semiconductor, Datasheet k MCU MC9S08DZ60 Rev. 4, 2008 URL: <http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08DZ60.pdf >

[4] Tým autorů Freescale Semiconductor, Datasheet k MCU MC9S08EL16 Rev. 3, 2008 URL: <http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08EL32.pdf >

[5] Tým autorů Freescale Semiconductor, Datasheet k LIN Physical Interface MC33399 Rev. 10, 2013 URL: <http://www.freescale.com/files/analog/doc/data_sheet/MC33399.pdf >

[6] ACARNELY, P. *Stepping Motors: A Guide to Theory and Practice (IEE Control Engineering Series 63)*. 4th ed. London(UK): the Institution of Electrical Engineers, 2002. 176 s. ISBN-10: 085296417X. ISBN-13: 978-0852964170.

[7] *LIN consortium.* [online] [ci. 2000-09-11]. URL:<http://www.lin-subbus.de>

[8] Tým autorů Freescale Semiconductor, Datasheet k motor driver MC33932 Rev. 5, 2012 URL: <http://www.freescale.com/files/analog/doc/data_sheet/MC33932.pdf >

[9] Pieš,M. *Regulace otáček stejnosměrného motoru*. Ostrava, 2006. Bakalářská práce na Fakultě elektrotechniky a informatiky VŠB-Technické univerzitě Ostrava na katedře měřicí a řídicí techniky. Vedoucí práce Jana Součková.

[10] Tým autorů Freescale Semiconductor, Datasheet k LIN Physical Interface MC33661 Rev.8, 2013 URL: <http://www.freescale.com/files/analog/doc/data_sheet/MC33661.pdf >

[11] Tým autorů Freescale Semiconductor, Datasheet k MCU MC9S08QG8 Rev. 5, 2009 URL: <http://www.freescale.com/files/microcontrollers/doc/data_sheet/MC9S08QG8.pdf >

[12] Tým autorů Future technology devicies: datasheet k FT232RL URL: <http://www.ftdichip.com/Support/Documents/DataSheets/ICs/DS_FT232R.pdf >

[13] Příklad zdrojového kódu pro LIN master a slave jednotku [online] <http://en.pudn.com/ > 2008

[14] Tým autorů POWERTIP, datasheet k LCD displayi POWERTIP 1602F <http://www.powertip.com.tw/products_2.php?product_id=1171043123&area_idbk=1170985616 >

[15] Freescale motor control – stepper motor [online] <http://www.freescale.com/webapp/sps/site/application.jsp?code=APLSTEMOT >

Přílohy

Příloha A – Master modul:

Cena výrobku				
název	typové značení	cena	účel	počet
IC1	7805 ONS	8	stablizátor Vcc	1
IC2	FT232RL	155,5	UART/USB	1
IC3	MC33399	17	LIN driver	1
IC4	MC9S08DZ60AMLC	265,5	MCU	1
iX1	Konektor 3 piny	12	konektor	1
iX2		6,7	USB konetkor typu A	1
BDM	BDM konektor	12	konektor	1
C1	C330n	1	keramický smd kondenzátor	1
C2,C6,C7,C9,C10	C100n	2,5	keramický smd kondenzátor	5
C3,4	C22p	1	keramický smd kondenzátor	2
C5,C8,C13	C10u	1,5	elektrolitický kondezátor	3
D1,2,3	1N4148 SMD	1,5	jistící dioda SMD	3
Q1	QM 16 MHZ	10	Krystal 16 MHZ	1
LED1	SMD LED GREEN	4,5	LED dioda	1
R1 – 4	R10K	2	odpory SMD	4
	Celkem [kč]	500,7	počet dílů	22

Deska plošného spoje Master modulu

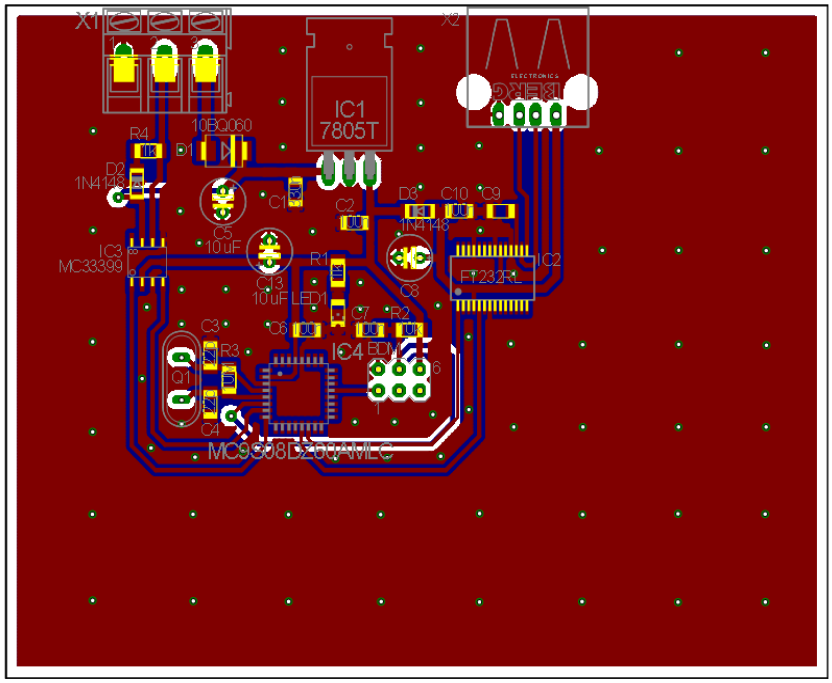
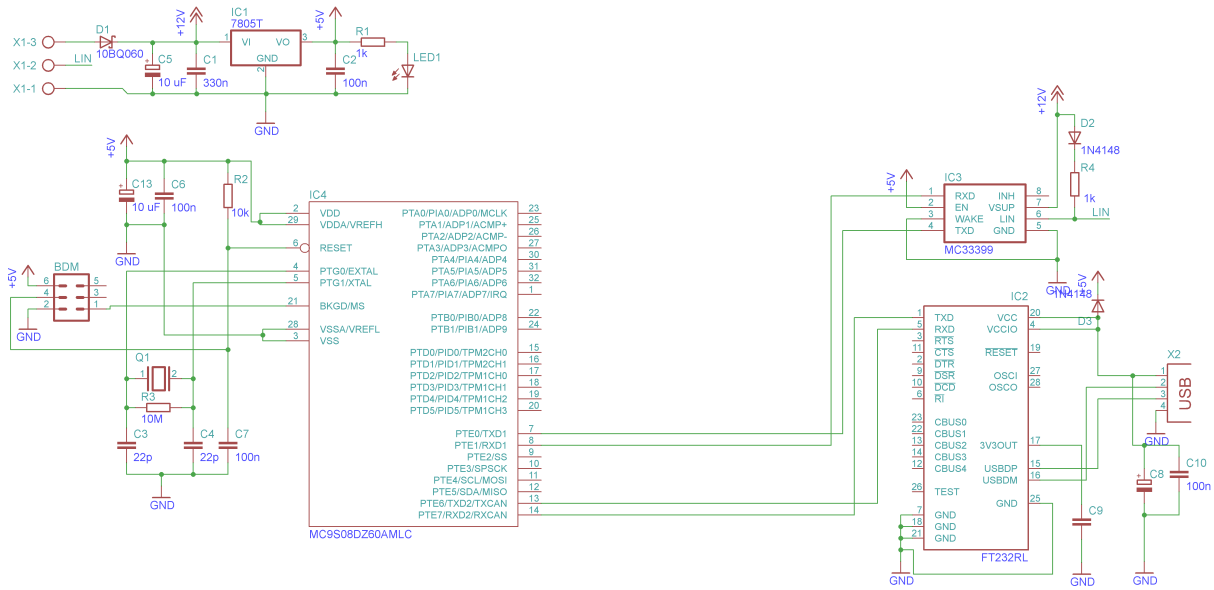


schéma zapojení modulu Master



Příloha B – LCD Modul:

Cena výrobku				
název	typové značení	cena	účel	počet
IC1	7805 ONS	8	stablizátor Vcc	1
IC2	MC33399	17	LIN driver	1
IC3	MC9S08EL16CTJ	91	MCU	1
DIS	POWERTIP PC1602F	112,3	LCD	1
iX1	Konektor 3 piny	12	konektor	1
BDM	BDM konektor	12	konektor	1
C1	C330n	1	keramický smd kondenzátor	1
C2,C6,C7	C100n	1,5	keramický smd kondenzátor	3
C3,4	C22p	1	keramický smd kondenzátor	2
C5,C8	C10u	1,5	elektrolitický kondezátor	2
D1	1N4148 SMD	1,5	jistící dioda SMD	1
Q1	QM 16 MHZ	10	Krystal 16 MHZ	1
LED1	SMD LED GREEN	4,5	LED dioda	1
R1 – 3	R10K	2	odpory SMD	3
R 4	R15K – PC1621NK010	11	potenciometr	1
	Celkem [kč]	286,3	počet dílů	17

Deska plošného spoje modulu LCD

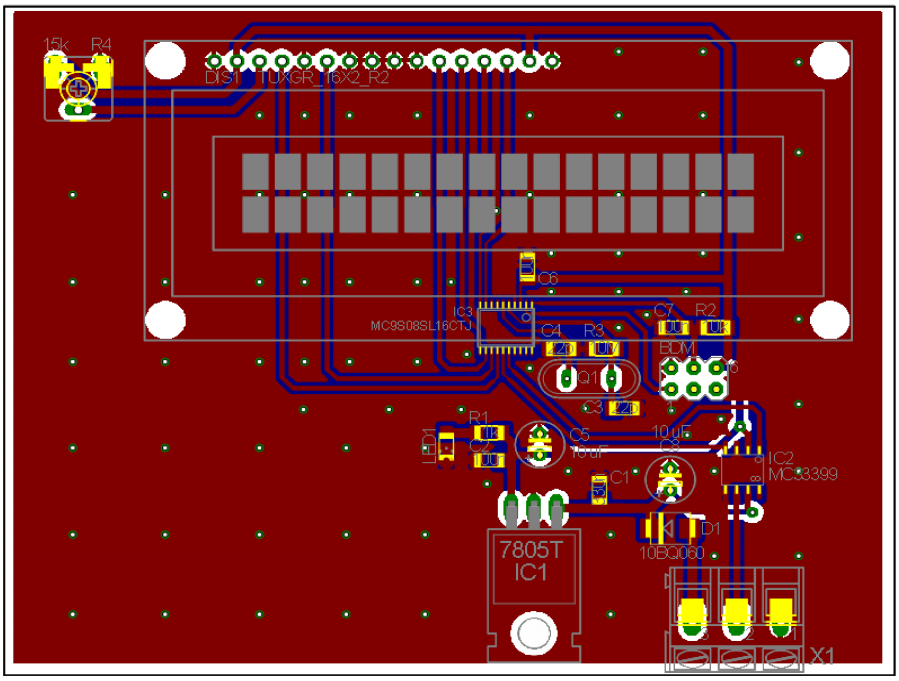
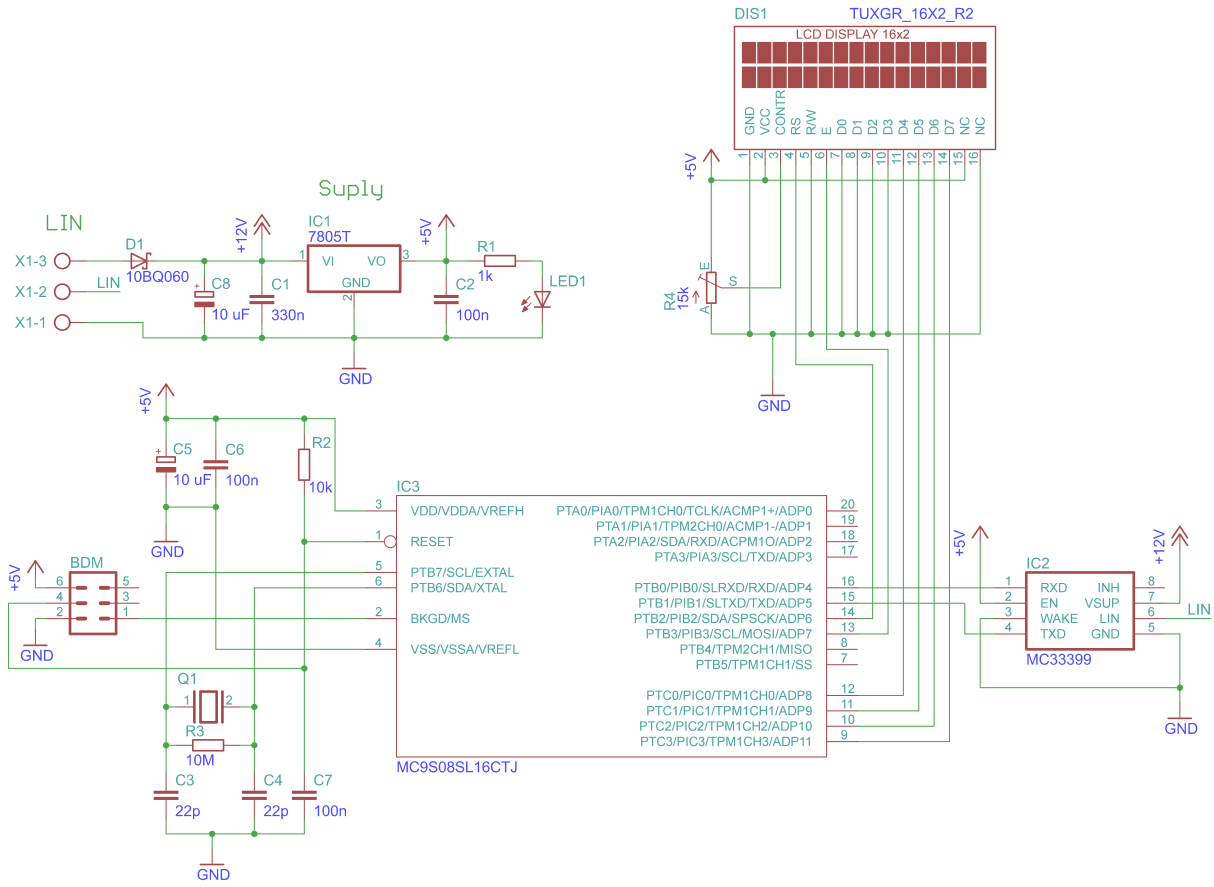


schéma LCD Slave Modulu



Příloha C – SS motor modul:

Cena výrobku

název	typové značení	cena	účel	počet
IC1	LF33CDT	32	stabilizátor Vcc	1
IC2	MC33932	155,5	driver pro řízení motoru	1
IC3	MC9s08QG8CTJ	42	MCU	1
IC4	MC33661	17	LIN driver	1
iX1	Konektor 3 piny	12	konektor	1
BDM	BDM konektor	12	konektor	1
JMP1,2	Konektor 2 piny	22,2	konektor	2
C2,6,7,10,11,12	C100n	2,5	keramický smd kondenzátor	5
C1	C330n	1	keramický smd kondenzátor	1
C3,4	C22p	1	keramický smd kondenzátor	2
C5,8,9,13	C100u	1	elektrolitický kondezátor	4
D1,2	1N4148 SMD	1,5	jistící dioda	3
Q1	QM 16 MHZ	10	Krystal 16 MHZ	1
LED1	SMD LED GREEN	4,5	LED dioda	1
R	R10K	2	odpory	4
M1	GRAUPER SPEED 500 E	299	ss motor	1
	Celkem [kč]	606	počet dílů	25

Schéma desky plošného spoje pro SS motor modul

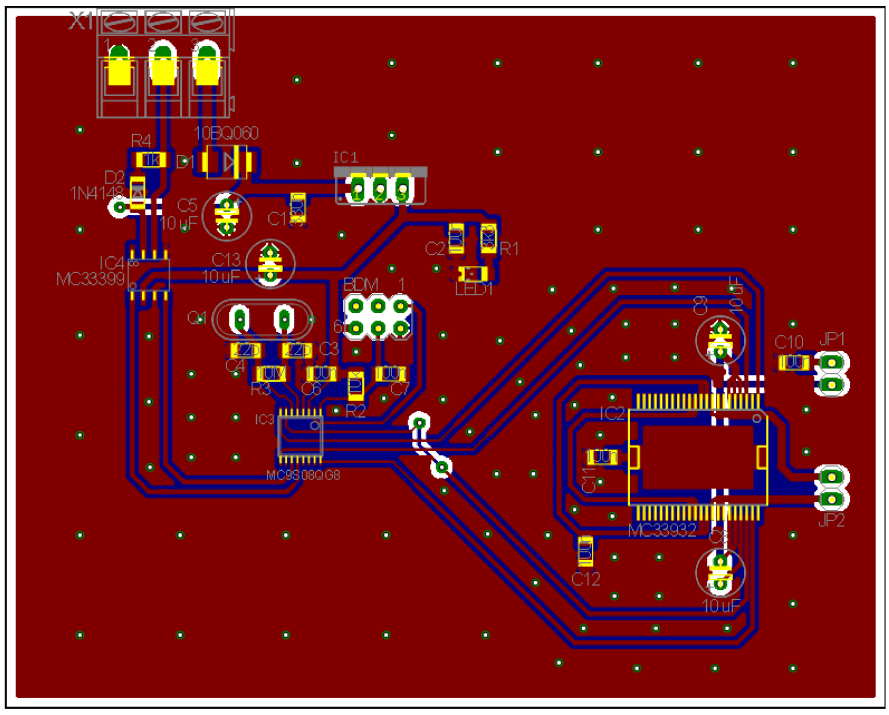
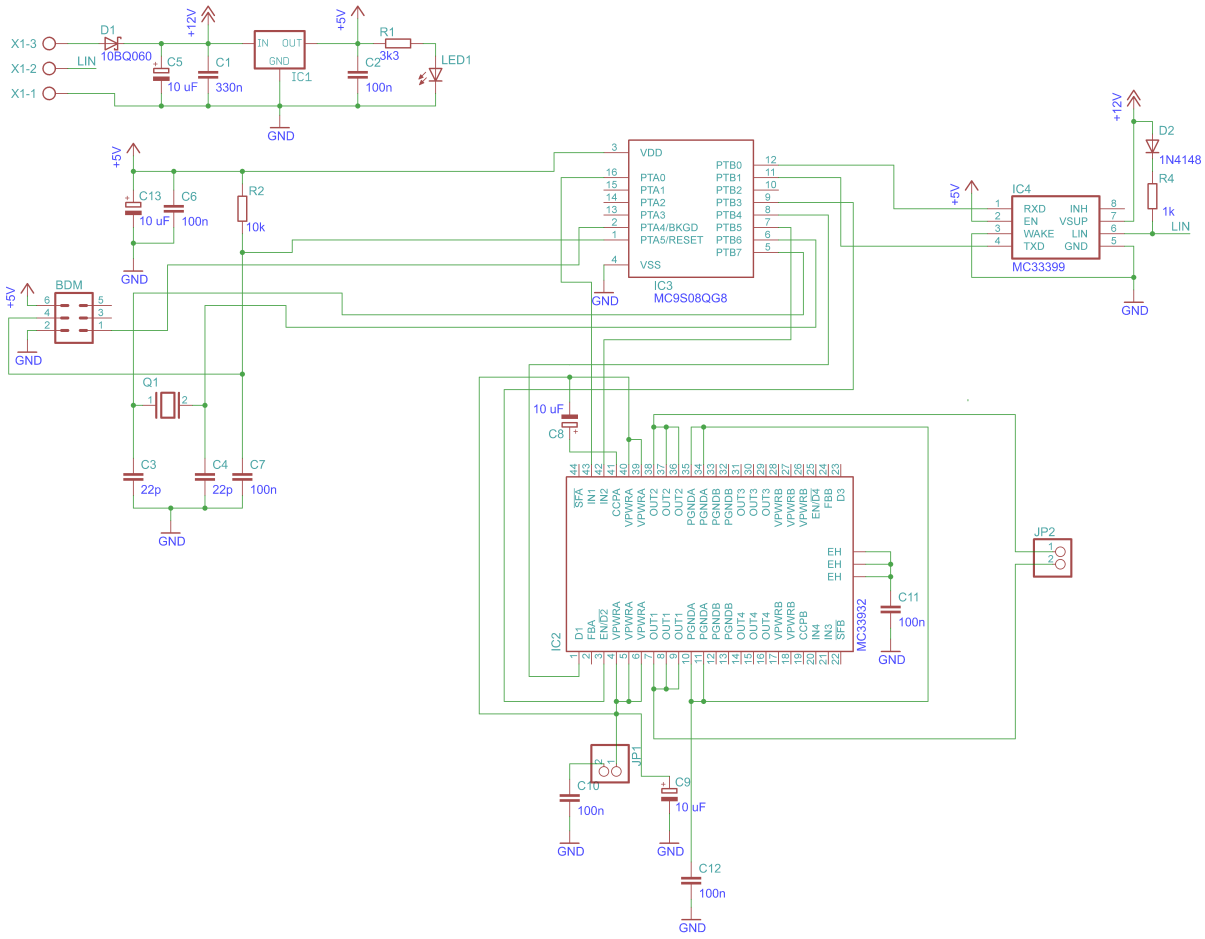


schéma zapojení modulu SS motor



Příloha D – krokový motor modul:

název	typové značení	cena	účel	počet
IC1	LF33CDT	32	stabilizátor Vcc	1
IC2	MC33932	155,5	driver pro řízení motoru	1
IC3	MC9s08QG8CTJ	42	MCU	1
IC4	MC33661	17	LIN driver	1
iV1	SN74AC04N	16,23	Invertor LV7404	1
iX1	Konektor 3 piny	12	konektor	1
BDM	BDM konektor	12	konektor	1
JMP1,2,3,5	Konektor 2 piny	44,4	konektor	2
C2,6,7 C10-16	C100n	4,5	keramický smd kondenzátor	9
C1	C330n	1	keramický smd kondenzátor	1
C3,4	C22p	1	keramický smd kondenzátor	2
C5,8,9,13	C100u	2	elektrolitický kondezátor	4
D1,2	1N4148 SMD	1,5	jistící dioda	3
Q1	QM 16 MHZ	10	Krystal 16 MHZ	1
LED1	SMD LED GREEN	4,5	LED dioda	1
R	R10K	2	odpory	4
M1	NANOTEC SP1518M0204	293	ss motor	1
	Celkem [kč]	606	počet dílů	25

Logická tabulka driveru MC33932, jež poukazuje na možnost realizace zpětného chodu i s méně vybaveným procesorem a s pomocí invertoru řídit driver jednoduše jako v případě procesoru s čtyřmi

Device State	Input Conditions				Status	Outputs	
	EN/$\overline{D2}$	D1	IN1	IN2	$\overline{SF}$	OUT1	OUT2
Forward	H	L	H	L	H	H	L
Reverse	H	L	L	H	H	L	H
Freewheeling Low	H	L	L	L	H	L	L
Freewheeling High	H	L	H	H	H	H	H
Disable 1 (D1)	H	H	X	X	L	Z	Z
IN1 Disconnected	H	L	Z	X	H	H	X
IN2 Disconnected	H	L	X	Z	H	X	H
D1 Disconnected	H	Z	X	X	L	Z	Z
Under-voltage Lockout(30)	H	X	X	X	L	Z	Z
Over-temperature(31)	H	X	X	X	L	Z	Z
Short-circuit(31)	H	X	X	X	L	Z	Z
Sleep mode EN/$\overline{D2}$	L	X	X	X	H	Z	Z
EN/$\overline{D2}$ disconnected	Z	X	X	X	H	Z	Z

kanály pro PWM stačí jen zadefinovat logic low PMW signál který se pohybuje kolem 1V a kde invertor má spodní hranici pro 0.8V je provedeno obrácení na logic High, který je roven 3V. Což bude mít za následek splnění podmínek pro reverzaci jelikož na IN1 a IN3 budou driverem registrovány v low úrovni ale IN2 a IN4 budou invertovány i s nosnou informací periody.

Deska plošného spojení pro modul krokového motoru

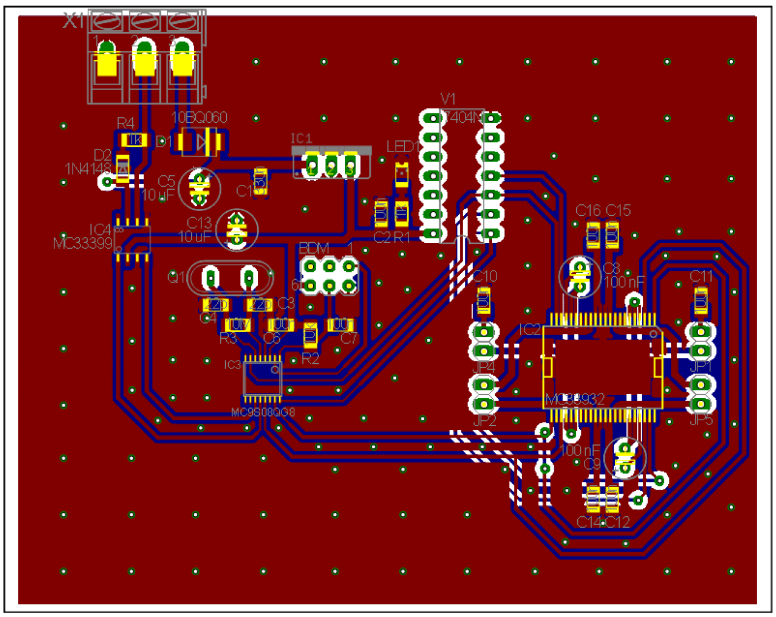


schéma zapojení modulu s krokovým motorem